Detection and Prediction of Errors in EPC Business Process Models

Jan Mendling
Institute of Information Systems and New Media
Vienna University of Economics and Business Administration (WU Wien)
Austria
jan.mendling@wu-wien.ac.at

22th of May 2007

A dissertation submitted to the
Institute of Information Systems and New Media,
Vienna University of Economics and Business Administration (WU Wien).

* * *

Dissertation

Detection and Prediction of Errors in EPC Business Process Models

Submitted to the Institute of Information Systems and New Media,
Vienna University of Economics and Business Administration, by

Jan
Mendling

in partial fulfillment of the requirements
for the degree of Doctor of Social Sciences and Economics
Under supervision of

Prof. Dr. Gustaf Neumann
Institute of Information Systems
and New Media
WU Wien

Prof. Dr. Wil M. P. van der Aalst
Department of Mathematics
and Computer Science
TU Eindhoven

Abstract

Business process modeling plays an important role in the management of business processes. As valuable design artifacts, business process models are subject to quality considerations. In this context, the absence of formal errors, such as deadlocks, is of paramount importance for the subsequent implementation of the process. This doctoral thesis provides a fourfold contribution to the understanding of such errors in business process models with a particular focus on Event-driven Process Chains (EPCs), a business process modeling language that is frequently used in practice. Firstly, we formalize the operational semantics of EPCs in a novel way so that matching OR-splits and OR-joins never deadlock. Secondly, and based on these semantics, we introduce a soundness criterion for EPCs that offers a precise identification of those models which have errors. For the verification of this soundness notion in practice, we present two analysis tools, a ProM plug-in for a verification based on the reachability graph, and a batch program called *xoEPC* for a verification based on reduction rules. Thirdly, we define a set of business process model metrics that are supposed to serve as predictors for error probability of an individual EPC. Fourthly, we use statistical methods and a sample of about 2000 EPCs from practice to derive a regression function for the prediction of error probability. This function is validated against a holdout set of 113 EPCs from textbooks showing that 90% of the EPCs could be classified correctly as having errors or not. These results emphasize the importance of quality issues in business process modeling and provides the foundations for innovations in tool support.

Zusammenfassung

Geschäftsprozessmodellierung spielt eine wichtige Rolle für das Management von Geschäftsprozessen. Als wertvolle Designartefakte sind Geschäftsprozessmodelle Gegenstand von Qualitätsbetrachtungen. In diesem Zusammenhang ist es von besonderer Wichtigkeit für die nachfolgende Implementierung des Prozesses, dass keine formalen Fehler wie etwa Verklemmungen in den Modellen vorhanden sind. Diese Doktorarbeit liefert vier wesentliche Beiträge zum Verständnis solcher Fehler in Geschäftsprozessmodellen. Das Augenmerk wird insbesondere auf Ereignisgesteuerte Prozessketten (EPKs) gelegt, da diese in der Praxis häufig benutzt werden. Zum Ersten formalisieren wir die operationale Semantik der EPK auf eine neue Art und Weise, sodass zusammengehörende ODER-Verzweigungen und ODER-Zusammenführungen niemals verklemmen. Zum Zweiten, und darauf aufbauend, stellen wir ein Korrektheitkriterium für EPKs vor, das eine präzise Identifikation von solchen Modellen ermöglicht, die Fehler enthalten. Für die praktische Verifizierung dieses Korrektheitskriteriums präsentieren wir

zwei Analysewerkzeuge, zum einen einen ProM-Plug-in zur Verifikation auf Basis des Erreichbarkeitsgraphens, und zudem ein Stapelverarbeitungsprogramm namens *xoEPC* zur Verifikation mit Hilfe von Reduktionsregeln. Zum Dritten definieren wir eine Menge von Geschäftsprozessmodellmetriken, die als Anzeiger für die Fehlerwahrscheinlichkeit einer einzelnen EPK dienen sollen. Zum Vierten benutzen wir statistische Methoden und eine Stichprobe von etwa 2000 EPKs aus der Praxis, um eine Regressionsfunktion zur Vorhersage von Fehlerwahrscheinlichkeiten abzuleiten. Diese Funktion wird anhand einer zweiten Stichprobe von 113 EPKs aus Lehrbüchern validiert, welche zeigt, dass 90% der EPKs richtig als fehlerhaft oder fehlerfrei klassifiziert werden konnten. Die Ergebnisse unterstreichen die Wichtigkeit von Qualitätsbetrachtungen in der Geschäftsprozessmodellierung und bieten eine Grundlage für Innovationen in der Werkzeugunterstützung.

Acknowledgement

On these pages it is time to thank all the people who were involved in this thesis, but who are not an author. I want to start with Prof. Dr. Gustaf Neumann, who served as a first supervisor, and Prof. Dr. Wil van der Aalst who, was second supervisor. In October 2003 I started working at Prof. Neumann's academic entity which is now called Institute of Information Systems and New Media (i.e. Institut für Wirtschaftsinformatik und Neue Medien). Prof. Neumann offered me a creative work environment where I could develop my ideas and academic writing skills. He was open and supportive with all matters, in particular, a change of the topic of my thesis, conference attendances, and research visits abroad. In 2004, I met Prof. van der Aalst at a conference and after several meetings at other conferences, he finally agreed to be my second supervisor. His feedback was extraordinarily helpful for the formalization of my concepts.

Additionally, I want to thank Prof. Dr. Markus Nüttgens for raising my interest in business process modeling and EPCs, and for his continuous support since my diploma thesis in Trier, Germany; Prof. Dr. Walter Schertler and Prof. Dr. Axel Schwickert for supporting my application in Vienna by serving as a reference; Prof. Dr. Carlos de Backer for being a role model in teaching Information Systems and explaining IT concepts in Dutch; and Prof. Dr. Hans Czap for his efforts in establishing the Diplomstudiengang Wirtschaftsinformatik at the University of Trier. Furthermore, I thank all my colleagues at the Institute of Information Systems and New Media and the Institute of Management Information Systems at the Vienna University of Economics and Business Administration (WU Wien) for providing a friendly work environment.

I was happy to collaborate with several excellent people on different paper projects: Wil van der Aalst, Paul Barborka, Alberto Brabenetz, Jorge Cardoso, Boudewijn van Dongen, Michael Hafner, Lukas Helm, Thomas Hornung, Cirano Iochpe, Georg Köldorfer, Agnes Koschmider, Kristian Bisgaard Lassen, Jean Michael Lau, Michael Moser, Michael zur Muehlen, Martin Müller, Gustaf Neumann, Markus Nüttgens, Cristian Pérez de Laborda, Andreas Pinterits, Adrian Price, Michael Rausch, Jan Recker, Manfred Reichert, Hajo Reijers, Michael Rosemann, Frank Rump, Bernd Simon, Carlo Simon, Guido Sommer, Mark Strembeck, Gerald Stermsek, Lucinéia Heloisa Thom, Irene Vanderfeesten, Eric Verbeek, Barbara Weber, Fridolin Wild, Uwe Zdun, and Jörg Ziemann. Thank you for sharing your insights with me. I also thank Kalina Chivarova for gathering some of the EPC models of the holdup sample in her bachelor thesis. Special thanks goes to Herbert Liebl, who had the nice idea to generate SVG graphics from the error analysis and highlight the errors in the EPC models.

Zuletzt einige Worte des Danks auf deutsch. Meinen Eltern danke ich für ihre nim-

mermüde Unterstützung und ihr Vertrauen in meine Fähigkeiten. Ohne ihren Rückhalt und den meiner Schwester, Omas und Opas, Tanten und Onkels, Ur-Omas und Ur-Opas und meiner Freunde wäre diese Arbeit nicht denkbar. Dies gilt insbesondere für Wiebke, mit der ich gemeinsam nach Wien gegangen bin. Zuletzt danke ich Leni. Sie gab mir die Geborgenheit und Wärme, die ich brauchte, um diese Arbeit zu einem guten Ende zu führen. Ich freue mich auf alles, was wir noch gemeinsam erleben werden.

Jan Mendling
April 2007

Contents

List of Figures

List of Tables

List of Acronyms

ACM	Association for Computing Machinery
AD	Anno Domini
AND	Logical and-operator
ARIS	ARchitecture of Integrated Information Systems
BPEL	Business Process Execution Language for Web Services
BPM	Business Process Management
BPMN	Business Process Modeling Notation
BWW	Bunge-Wand-Weber
CASE	Computer Aided Software Engineering
CC	Cyclomatic complexity, cyclomatic number
CCD	Cyclomatic complexity density
CDIF	CASE Data Interchange Format
C-EPC	Configurable Event-driven Process Chain
CFC	Control-flow complexity
COCOMO	Constructive Cost Model
CPN	Colored Petri Nets
DOM	Document Object Model
ebXML	Electronic Business XML
ECC	Essential cyclomatic complexity
EPC	Event-driven Process Chain
EPK	Ereignisgesteuerte Prozesskette
EPML	Event-driven Process Chain Markup Language
ERD	Entity-relationship diagram
ERP	Enterprise Resource Planning
ET	Entscheidungstabellen-operator
FSM	Finite State Machine
GHz	Giga-Hertz
GI	Gesellschaft für Informatik e.V., German Informatics Society
GoM	Guidelines of Modeling

GQM	Goal-Question-Metric
GXL	Graph Exchange Language
HTML	Hypertext Markup Language
ID	Identifier
IEC	International Electrotechnical Commission
IEEE	Institute of Electrical and Electronics Engineers
IFC	Information Flow Complexity
ISO	International Organization for Standardization
IT	Information technology
IWi	Institut für Wirtschaftsinformatik
KIM	Kölner Integrationsmodell
KOPeR	Knowledge-based organizational process redesign
LoC	Lines of Code
LTL	linear temporal logic
MI	Multiple Instantiation
modEPC	Modified Event-driven Process Chain
OASIS	Organization for the Advancement of Structured Information Standards
oEPC	Object-oriented Event-driven Process Chain
OR	Logical or-operator
ProM	Process Mining tool of TU Eindhoven
rEPC	Real-time Event-driven Process Chain
RG	Reachability Graph
SAP	Systeme, Anwendungen und Produkte in der Datenverarbeitung
SCOOP	System for Computerization of Office Processing
SEQ	Sequence-operator
SNA	Social network analysis
SPSS	Statistical Package for the Social Sciences
SQL	Structured Query Language
SVG	Scalable Vector Graphics
TAM	Technology acceptance model
Tcl	Tool command language
tDOM	Tool command language Document Object Model
UML	Unified Modeling Language
UML AD	Unified Modeling Language Activity Diagram
US	United States
Wf. nets	Workflow nets
WfMC	Workflow Management Coalition
Woflan	WOrkFLow ANalyzer

WS-CDL	Web Services Choreography Description Language
xEPC	Agent-oriented Event-driven Process Chain
XHTML	Extensible Hypertext Markup Language
XML	Extensible Markup Language
xoEPC	Extended Object Event-driven Process Chain
XOR	logical exclusive or-operator
XOTcl	Extended Object Tool command language
XPDL	Extensible Markup Language Process Definition Language
YAWL	Yet Another Workflow Language
yEPC	Yet another Event-driven Process Chain

Chapter 1

Introduction

This chapter provides an introduction to this doctoral thesis. After a discussion of the general motivation in Section 1.1, we present the research contribution of this thesis in Section 1.2. In Section 1.3, we discuss the findings from an epistemological point of view and relate them to design science and behavioral science approaches to information systems research. Finally, Section 1.4 closes this chapter with an outlook on the structure of this thesis.

1.1 Motivation

The importance of Business Process Management (BPM) is reflected by the figures of the related industry. For example, Wintergreen Research estimates that the international market for BPM-related software and services accounted for more than 1,000 million US dollars in 2005 with a tendency of rapid growth in the next couple of years [Win06]. Furthermore, the plethora of popular and academic textbooks (e.g. [HC93,

Dav93, JB96, Sch98a, ACD+99, Sch00, LR00, ADO00, AH02, MCH03, BKR03, Kha04, LM04, Hav05, SF06, Sta06, Jes06, SCJ06, WLPS06, KRM06, Smi07]) as well as international professional and academic conference series, such as the BPM conference [AHW03, DPW04, ABCC05, DFS06], confirm the importance of BPM. Despite the overall recognition of its importance, several fundamental problems remain unsolved by current approaches.

A particular problem in this context is the lack of research regarding what is to be considered good design. The few contributions in this area reveal an incomplete understanding of quality aspects in this regard. Business process modeling as a sub-discipline of BPM faces a particular problem. Often, modelers who have little background in formal methods, design models without understanding the full implications of their specification (see e.g. [PH07]). As a consequence, process models designed on a business level can hardly be reused on an execution level since they often suffer from formal errors, such as deadlocks.[1] Since the costs of errors increase exponentially over the development life cycle [Moo05], it is of paramount importance to discover errors as early as possible. A large amount of work has been conducted to try to cure the symptoms of this weak understanding by providing formal verification techniques, simulation tools, and animation concepts. Still, several of these approaches cannot be applied since the business process modeling language in use is not specified appropriately. Furthermore, this stream of research does not get to the root of the problem. As long as we do not understand why people introduce errors in a process model, we will hardly be able to improve the design process. There is some evidence on error rates for one particular collection of business process models from practice [MMN+06b, MMN+06c].[2] We will take this research as a starting point to contribute to a deeper understanding of errors in business process models.

[1] In the subsequent chapters, we will distinguish between two major types of errors. Firstly, formal errors can be identified algorithmically with verification techniques. Secondly, inconsistencies between the real-world business process and the process model can only be detected by talking to stakeholders. The focus of this thesis will be on formal errors.

[2] Classroom experiences are reported, for example, in [MSBS02, Car06].

1.2 Research Contributions

The research objective of this doctoral thesis is the development of a framework for the detection of formal errors in business process models, and the prediction of error probability based on quality attributes of these models. We will focus on Event-driven Process Chains (EPCs), a business process modeling language that is heavily used in practice. The advantage of this focus is, firstly, that the results of this thesis are likely to have an impact on current modeling practices. Secondly, there is a large empirical basis for analysis. By tapping the extensive stock of EPC model collections, we aim to bring forth general insights into the connection between process model attributes and error probability. In order to validate such a connection, we first need to establish an understanding of model attributes that are likely connected with error probability. Furthermore, we must formally define an appropriate notion of correctness, which gives an answer to the question whether a model has a formal error or not. It is a prerequisite to answering this question that we define the operational semantics of the process modeling language, i.e. EPCs, in a formal way. Against the state of the art, this thesis provides the following technical contributions.

Formalization of the OR-join: The semantics of the OR-join have been debated for more than 10 years now. Existing formalizations suffer either from a restriction of the EPC syntax (see e.g. [CS94, LA94, LSW98, Aal99, DR01]) or from non-intuitive behavior (see e.g. [NR02, Kin06, AH05, WEAH05]). In Chapter 3 of this thesis we formalize the EPC semantics concept as proposed in [MA06]. In comparison to other approaches, this novel formalization has the advantage that it is not restricted to a subset of EPCs, and that it provides intuitive semantics for blocks of matching OR-splits and joins since they cannot deadlock. The calculation of the reachability graph was implemented as a plug-in for ProM as a proof of concept. In this way, this novel semantics definition contributes to research on the specification of business process modeling languages.

Verification of process models with OR-joins and multiple start and end events: Verification techniques for process models with OR-joins and multiple start and end events suffer from one of two problems. Firstly, they build on an approxima-

tion of the actual behavior and, therefore, do not provide a precise answer to the verification problem, e.g. by considering a relaxed notion of soundness [DR01], by involving user decisions [DAV05], or by approximating relaxed soundness with invariants [VA06]. Secondly, there are verification approaches for semantics definitions (see [CFK05, WAHE06]) that suffer from the previously mentioned non-intuitive behavior. While this is not a problem of the verification itself, all these approaches are not tailored to cope with multiple start and end events. In Chapter 4 of this thesis, we specify a dedicated soundness criterion for EPC business process models with OR-joins and multiple start and end events. Furthermore, we define two verification approaches for EPC soundness, one as an explicit analysis of the reachability graph, and a second based on reduction rules to provide a better verification performance. Both approaches were implemented as a proof of concept. In this way, we contribute to the verification of process models with OR-joins and multiple start and end events, and in particular, we extend the set of reduction rules for business process models.

Metrics for business process models: Metrics play an important role in the operationalization of various quality-related aspects in software engineering, network analysis, and business process modeling. Several authors use metrics to capture different aspects of business process models that are presumably related to quality (see e.g. [LY92, Nis98, Mor99, RV04, Car05d, BG05, CGP+05, CMNR06, LG06, ARGP06c, MMN+06b, MMN+06c]). A problem of these works is that business process-specific concepts like sequentiality, decision points, concurrency, or repetition are hardly considered, and too often simple count metrics are defined. Furthermore, there appears to be little awareness of related research, maybe because process model measurement is conducted in separate disciplines including software process management, network analysis, Petri nets theory, and conceptual modeling. In Chapter 5 of this thesis, we will provide an extensive list of metrics for business process models and relate it to previously isolated research. Beyond that, we provide a detailed discussion of the rationale and the limitations of each metric, which is meant to serve as a predictor for error probability. We formulate a hypothesis for each metric based on whether it is positively or negatively correlated

with error probability.

Validation of metrics as error predictors: Up to now, there is little empirical evidence for the validity of business process model metrics as predictors for error probability. Some empirical work was conducted, but with a different focus. *Lee and Yoon* investigate the empirical relationship between parameters of Petri nets and their state space [LY90, LY92]. *Canfora et al.* empirically evaluate the suitability of metrics to serve as predictors for maintainability of the process model [CGP+05]. *Cardoso* analyzes the correlation between the control flow complexity metric with the perceived complexity of process models [Car06]. Most related to this thesis is an analysis of the SAP Reference Model where *Mendling et al.* test a set of simple count metrics as error predictors [MMN+06b, MMN+06c]. In Chapter 6 of this thesis, we use logistic regression for the test which is similar to the analysis of the SAP Reference Model. Still, we consider both the broader set of metrics from Chapter 5, a precise notion of EPC soundness as defined in Chapter 4, and a much broarder sample of EPC models from practice. The results do not only show that certain metrics are indeed a good predictor for error probability, but also that simple count metrics fail to capture important aspects of a process model.

Little research in information systems tries to combine design science and behavioral science research paradigms (see e.g. [BH05]). Since the previously listed contributions cover both design and behavioral aspects, we consider the main contribution of this thesis to be the innovative and holistic combination of both these research paradigms in order to deliver a deeper understanding of errors in business process modeling.

1.3 Epistemological Position

This thesis contributes to information systems research as defined by *Hevner, March, Park, and Ram* [HMPR04]. It covers different research aspects that build on both design science and behavioral science paradigms. Section 1.3.1 introduces the Information Systems Research Framework as presented by *Hevner, March, Park, and Ram* [HMPR04], that overarches design science and behavioral science in information systems research.

Section 1.3.2 uses the information systems research guidelines to discuss in how far this thesis meets information systems research standards.

1.3.1 Information Systems Research Framework

Information systems research is the study of phenomena related to information systems. Information systems research and its German counterpart Wirtschaftsinformatik build on both design science and behavioral science research. According to *Hansen and Neumann* [HN05], it is defined as follows: "The study that is concerned with design of computer-based information systems in business is called Wirtschaftsinformatik (in English: Management Information Systems, Information Systems, Business Informatics). It is understood to be an interdisciplinary subject between business science and computer science" (my translation).[3] This definition stresses the design science paradigm which is typical for the European information systems community [BH05, BN07], but it also covers behavioral aspects related to design. Only recently, there has been a trend to widen the scope of Wirtschaftsinformatik by taking advantage of more behavioral, especially empirical, methodologies [BH05].

Behavioral science "seeks to develop and justify theories [...] that explain or predict organizational and human phenomena surrounding the analysis, design, implementation, management, and use of information systems" [HMPR04, p.76]. A typical example of a theory that follows a behavioral science paradigm is the technology acceptance model (TAM) [Dav89]. According to the TAM, user acceptance of information technology can be explained by two major factors: perceived usefulness and perceived ease of use. Since information systems are created by making design decisions, such insights into behavioral aspects provide feedback for the design of new artifacts.

The foundations of information systems research as a *design science* were elaborated in the seminal work of *Simon* on the *Sciences of the Artificial* [Sim96]. In this context, design science is understood as a problem-solving process. A key characteristic of prob-

[3] Similar definitions are given by *Mertens, Bodendorf, König, Picot and Schumann* [MBK+98], *Stahlknecht and Hasenkamp* [SH05], *Ferstl and Sinz* [FS98], *Heinrich, Heinzl, and Roithmayr* [HHR07], or *Lehner* [Leh97].

lems in design science is *wickedness*, i.e. there is no definitive formulation of the problem due to unstable requirements, ill-defined environmental context, complex interactions, inherent change, and of psychological and social factors being involved (cf. [HMPR04]). Therefore, the solution cannot be assessed by truth, but rather by utility. Based on the assumption of bounded rationality of a human as a problem-solver, *Simon* advocates to accept satisficing solutions by designing and creating useful artifacts. In information systems research, design science relates to building and evaluating design artifacts including constructs, models, methods, and instantiations (cf. [MS95]). These artifacts facilitate the exploration of the space of design choices [BBC+04]. Information systems design theories prescribe effective development practices that can be applied for a particular class of user requirements to construct a certain type of system solution [MMG02]. The created information systems artifacts influence and extend the capabilities of organizations and human problem-solving, i.e. they establish a new reality. Respective theories on their application and impact are expected to follow their development and use [HMPR04].

While "behavioral science addresses research through the development and justification of theories that explain or predict phenomena related to the identified business need," design science "addresses research through the building and evaluation of artifacts designed to meet the identified business need. The goal of behavioral-science research is truth. The goal of design-science research is utility" [HMPR04]. The assessment of artifacts (evaluation) or theories (justification) can lead to the identification of weaknesses. Such insight can be used for refinement of artifacts and theories. The research design of this thesis combines both paradigms following the concept of *Hevner et al.* that design and behavioral science are complementary: "truth informs design and utility informs theory" [HMPR04].

A key characteristic of *information systems* in organizations is that they are utilized for "improving the effectiveness and efficiency of that organization" [HMPR04, p.76]. Accordingly, the overall goal of information systems research can be defined as to "further knowledge that aids in the productive application of information technology to human organizations and their management" [ISR02]. Thus, information systems research is conducted in an *environment* that involves people, organizations, and technology in order to enhance the *knowledge base* of foundations and methodologies in this area (cf.

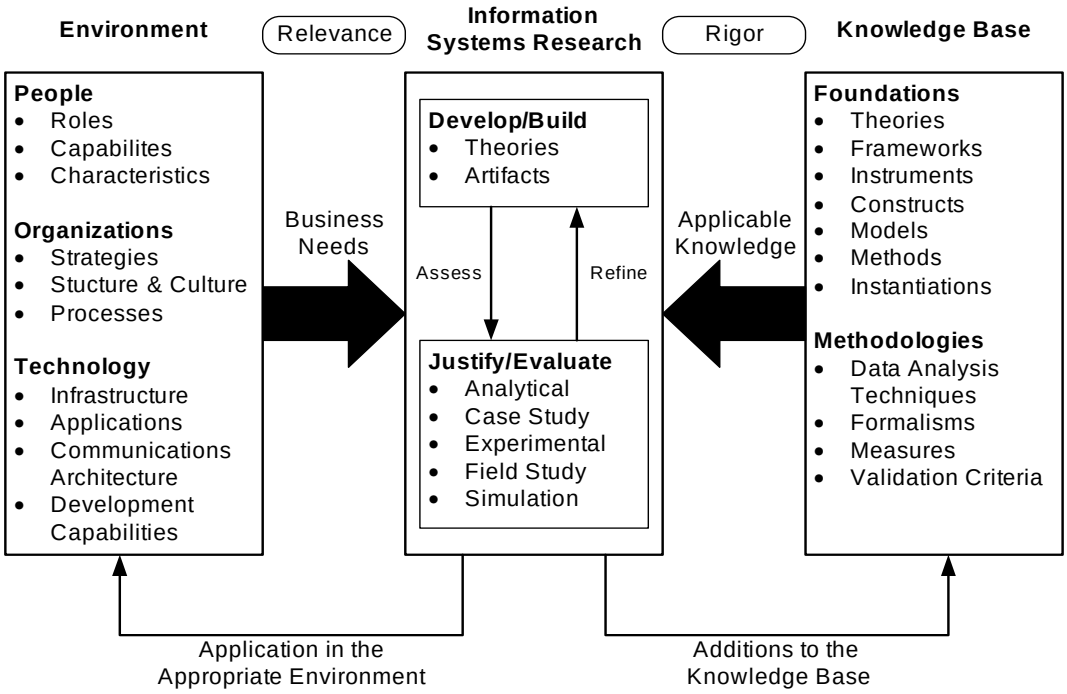


Figure 1.1: Information Systems Research Framework as defined by Hevner et al. [HMPR04, p.80]

Figure 1.1). Due to the involvement of people and organizations, such knowledge can be acquired following two different research paradigms: behavioral science and design science. Both build on a creative activity of developing theories or building artifacts, respectively, and an analytical activity of justification or evaluation, respectively (see Figure 1.1).

The *environment* of information systems research includes those entities that define the problem space, i.e. people, organizations, and technology. It defines the background against which business needs emerge. These business needs are influenced by the roles, capabilities, and characteristics of people, and shaped in consideration of an organization's strategy, structure, culture, and business processes. Moreover, business needs reflect current and prospective technology such as infrastructure, applications, communications architecture, and development capabilities. The researcher aligns her problem perception to these factors in order to establish *relevance*.

The *knowledge base* offers solutions to problems which are already well understood. Development and building can rely on foundations like theories, frameworks, instruments, constructs, models, methods, and instantiations that have resulted from prior research. Methodologies like data analysis techniques, formalisms, measures, and validation criteria are valuable in the justification and evaluation phase. The researcher applies existing foundations and methodologies to a given problem in order to establish *rigor*. Behavioral science often considers empirical evidence, while design sciences tends to use mathematical methods more frequently. The overall goal of both behavioral and design science is to address the business need and to contribute to the knowledge base for future application. The lack of addition to the knowledge base can be used to distinguish routine design and design research. While routine design tackles business needs by applying existing knowledge, design research establishes either innovative solutions to unsolved problems or more efficient or effective solutions to solved problems. Accordingly, design research contributes to the knowledge base while routine design does not.

1.3.2 Relation to Information Systems Research Guidelines

The Information Systems Research Framework emphasizes the similarities between behavioral science and design science. Related to that, *Hevner et al.* suggest a set of seven guidelines for effective information systems research, in particular for works with a design science focus [HMPR04]. On the following pages, we use these guidelines to discuss in how far this thesis meets information systems research standards.

Guideline 1: Design an Artifact Information systems research aims to design purposeful artifacts addressing business needs within an organizational setting. Artifacts in this context include constructs, models, methods, and instantiations [HMPR04]. In this thesis, we formalize EPC semantics, EPC soundness, and error metrics as constructs that can be used to analyze and simulate EPC business process models. Furthermore, we define methods in this sense to calculate the reachability graph, to verify soundness based on reachability graph analysis as well as reduction rules, and to calculate metrics from the process models. Finally, we present prototypical

implementations of these methods (i.e. instantiations) as a plug-in for ProM and as a software component called xoEPC in order to demonstrate feasibility.

Guideline 2: Problem Relevance Relevance of information systems research is constituted by addressing a problem of development or practical application of information systems; and in particular, their planning, management, design, operation, and evaluation [HMPR04]. The general business need of this research stems from a wide-spread application of business process management in practice, and of EPCs as a modeling language in particular. The findings and concepts presented in this thesis contribute to several aspects of quality assurance in business process modeling.

Guideline 3: Design Evaluation The utility of an artifact in a given problem situation must be clearly established using evaluation methods [HMPR04]. The completeness and the correctness of the EPC semantics definition and the soundness analysis is checked using analytical methods. The usefulness of business process model metrics is first evaluated in a descriptive way before using statistical methods. The implementations of the verification methods were extensively tested with numerous EPC models.

Guideline 4: Research Contribution The design research has to provide a novel, significant, and general contribution to the knowledge base; otherwise it has to be considered as design routine [HMPR04]. The contributions have already been presented in Section 1.2. They include a novel formalization of the OR-join (design science), two verification approaches for process models with OR-joins and multiple start and end events (design science), metrics for business process models (design science), and a validation of the metrics as predictors of error probability using an extensive set of EPC business process models from practice (behavioral science).

Guideline 5: Research Rigor Rigor refers to the way in which construction and evaluation of design science is conducted. This implies that the researcher has to effectively make use of the knowledge base and its methodologies and foundations [HMPR04]. For this thesis, we took advantage of prior research on business pro-

cess modeling languages, predicate logic, formal semantics, graph theory, software measurement, and logistic regression.

Guideline 6: Design as a Search Process Problem solving in design science can be defined as utilizing suitable means to reach desired ends while respecting laws imposed by the environment [Sim96]. Suitable means in this context refer to an available operation that can be used to build a solution, ends represent goals and constraints, and laws capture forces of the environment that cannot be controlled. The wickedness of the design-science problem implies that means, ends, and laws cannot be represented on the level of completeness and precision that would be needed for an optimization problem. The problem of finding predictors for error probability in business process models exactly displays this wickedness. In this thesis, we seek to establish a satisficing solution in the terms of *Simon*, based on a set of business process model metrics and on a notion of correctness called EPC soundness. In this setting, it is crucial to demonstrate that a certain solution *does* work, even if it is not yet completely understood *why* it works (cf. [HMPR04]). Using a logistic regression approach, we are not only able to show that this set of metrics does suit for predicting errors, but also that the hypothetical direction of the impact can be validated and that it outperforms existing approaches.

Guideline 7: Communication of Research The design solution has to be presented to both the academic community and to practitioners who might be interested in the findings [HMPR04]. For the research community, communication extends the knowledge base and offers repetition of research in order to check for correctness. Working on this thesis has led to the publication of five journal articles, five book chapters, 49 workshop and conference papers, and 19 technical reports and popular publications. This means that several concepts of this thesis are already publicly available as part of the information systems knowledge base.

Relating this thesis to the information systems research guidelines highlights that it suffices international research standards in this discipline and that it enhances its knowledge base in several directions.

1.4 Structure of this Thesis

This thesis is organized in seven chapters. It starts with a general overview of business process management, continues with semantics of Event-driven Process Chains and the verification of soundness before discussing metrics for business process models that are subsequently validated for their capability to predict error probability.

Chapter 1: Introduction In this chapter, we sketch the motivation of this thesis, present its contributions, and discuss its epistemological position related to information systems research.

Chapter 2: Business Process Management This chapter discusses the backgrounds of business process management and defines important terms related to it. Furthermore, it sketches the importance of business process modeling and the role of errors in the business process management lifecycle.

Chapter 3: Event-driven Process Chains (EPC) This chapter gathers state-of-the-art work on EPCs. Building on the foundations of prior work, we establish a novel syntax definition and a novel semantics definition for EPCs. Our semantics are-based on transition relations that define both state changes and context changes. Furthermore, we present an algorithm to calculate the reachability graph of an EPC based on the transition relations and a respective implementation as a plug-in for ProM. The major motivations for this novel semantics are, firstly, semantic gaps and, secondly, non-intuitive behavior of existing formalizations.

Chapter 4: Verification of EPC Soundness This chapter presents an EPC-specific version of soundness as a criterion of correctness for EPCs. We propose two different approaches for the verification of soundness, one based on the reachability graph and another based on reduction rules. While the first approach explicitly considers all states and transitions that are represented by an EPC, there is a problem with state explosion, as the maximum number of states grows exponentially with the number of arcs. In order to avoid a performance problem, we introduce a set of reduction rules as second approach. This set extends prior work with new reductions for start and end components, delta components, prism components, and homoge-

neous EPCs. The second approach is tested by reducing the SAP Reference model. It shows that the reduction approach is *fast*, that it provides a *precise* result for almost all models, and that it finds *three times as many errors* as other approaches based on relaxed soundness.

Chapter 5: Metrics for Business Process Models This chapter discusses the suitability of business process model metrics to predict error probability from a theoretical point of view. Revisiting related research in the area of network analysis, software measurement, and metrics for business process models, we find that several aspects of process models are not yet combined in an overall measurement framework. Based on theoretical considerations, we present a set of 15 metrics related to size and 13 metrics that capture various aspects of the structure and the state space of the process model. For each of the metrics, we discuss their presumable connection with error probability and formulate respective hypotheses.

Chapter 6: Validation of Error Metrics In this chapter, we conduct several statistical analyses related to the connection between metrics and error probability. The results of the correlation analysis and the logistic regression model strongly confirm the hypothetical impact direction of the metrics. Furthermore, we derive a logistic regression function, based on a sample of about 2000 EPC business process models from practice, that correctly classifies 90% of the models from a second independent sample.

Chapter 7: Conclusions This chapter summarizes the findings and offers an outlook on future research. In particular, we discuss the implications of this thesis for guidelines and management for the business process modeling process, respective tool support, EPCs as a business process modeling language, and teaching of business process modeling.

Chapter 2

Business Process Management

This chapter provides an overview of business process management and business process modeling. Section 2.1 elaborates on the background of business process management by giving a historical classification of seminal work. Section 2.2 defines business process management and illustrates it by the help of the business process management life cycle. Business process models play an important role in this life cycle. Section 2.3 discusses modeling from a general information systems point of view and deducts a definition for business process modeling. Section 2.4 details business process modeling and distinguishes between formal verification and external validation of business process models. Furthermore, it emphasizes the need to understand why formal errors are introduced in business process models. Finally, Section 2.5 concludes the chapter with a summary.

2.1 History of Business Process Management

In the last couple of years, there has been a growing interest in business process management, from practice as well as from business administration and information systems

research. In essence, business process management deals with the efficient coordination of business activities within and between companies. As such, it can be related to several seminal works on economics and business administration. *Henri Fayol* as one of the founders of modern organization theory recommended a *subdivision of labor* in order to increase productivity [Fay66, p.20]. Already *Adam Smith* illustrated its potential benefits by analyzing pin production [Smi76]. As a drawback, subdivision of labor requires *coordination* between the subtasks. Business process management is concerned with coordination mechanisms, in order to leverage the efficient creation of goods and services in a production system based on such subdivision of labor. In this context, the *individual tasks* and the *coordination between them* can be subject to optimization efforts. *Frederick Taylor* advocated the creation of an optimal work environment based on scientific methods to leverage the most efficient way of performing individual work steps. In the optimization of each step, he proposed to "select the quickest way", to "eliminate all false movements, slow movements, and useless movements", and to "collect into one series the quickest and best movements" [Tay11, p.61]. The efficient coordination of business processes is addressed by the innovation of the assembly line system. Its inventor *Henry Ford* proudly praised the production cycle of only 81 hours in his company "from the mine to the finished machine" to illustrate the efficiency of the concept [For26, p.105].

In academia, *Nordsieck* was one of the first to distinguish structural and process organization [Nor32, Nor34]. He described several types of workflow diagrams (Ablaufschaubilder), e.g. for subdivision and distribution of labor, sequencing of activities, or task assignment [Nor32]. In this context, he identifies the order of work steps and the temporal relationship of tasks as the subject of process analysis with the overall goal of integrating these steps [Nor34]. He distinguishes between five levels of automation: free course of work, concerning the contents bound course of work, concerning the order bound course of work, temporally bound course of work, and concerning the beat bound course of work [Nor34].

In the decades after World War II, organization research (at least in German speaking countries) devoted more attention to structural organization than to process organization (cf. e.g. [Ulr49, Kos62, Gro66]). In the early 1970s, it became apparent that information systems would become a new design dimension in an organizational setting (cf. e.g.

Grochla or *Hansen* [Gro66, Han70, Gro75, GS75]). But the focus, even in this context, remain on the structure. The early books by *Scheer* [Sch76, Sch78, Sch84] nicely illustrate the focus on database technology in order to support business functions without giving much attention to process organization. At that time, the logic of business processes used to be hard-coded in applications such as production floor automation systems and were, therefore, difficult to change [HK96, Mue04]. Office automation technology during the late 1970s was the starting point for a more explicit control over the flow of information and the coordination of tasks. The basic idea was to build electronic forms for clerical work that was originally handled via paper. In his doctoral thesis, *Zisman* [Zis77, Zis78] used Petri nets [Pet62] to specify the clerical work steps of an office agent and introduced a respective prototype system called SCOOP. A comparable approach was presented by *Ellis* [Ell79], who modelled office procedures as Information Control Nets, a special kind of Petri nets consisting of activities, precedence constraints, and information repositories. An overview of further work on office automation is provided in [EN80].

Although the business importance of processes received some attention in the 1980s (e.g. [Por85]) and new innovations were introduced in information system support of processes (e.g. system support for communication processes [Win88] based on speech act theory introduced by [Aus62, Sea69]), it was only in the early 1990s that workflow management prevailed as a new technology to support business processes. An increasing number of commercial vendors of workflow management systems benefited from new business administration concepts and ideas such as process innovation [Dav93] and business process reengineering [HC93]. On the other hand, these business programs heavily relied on information system technology, in particular workflow systems, in order to establish new and more efficient ways of doing business. In the 1990s, the application of workflow systems, in particular those supporting information systems integration processes, profited from open communication standards and distributed systems technology that both contributed to interoperability with other systems (cf. [GHS95]). The Workflow Management Coalition (WfMC) founded in 1993 is of special importance for this improvement [Hol04]. The historical overview of office automation and workflow systems given in [Mue04, p.93] nicely illustrates this breakthrough (see Figure 2.1). This

period also saw a growing body of scientific publications on workflow technology and process specification (see e.g. [EN93, GHS95, CCPP95, JB96, SRS96, OS96, Obe96, RD98, Aal98, Wes00, LR00]). Up to the late 1990s, intra-enterprise processes remained the major focus of business process management [DHL01].

Since the advent of the eXtended Markup Language (XML) and web services technology, application scenarios for business process integration have become much easier to implement in an inter-enterprise setting. Current standardization efforts mainly address interoperability issues related to such scenarios (cf. e.g. [MNN04, MNN05b, MMP05]). The common interest of the industry, to facilitate the integration of interorganizational processes, leverages the specification of standards for web service composition, like the Business Process Execution Language for Web Services (BPEL) [CGK+02, ACD+03, AAB+05], for web service choreography, like the Web Service Choreography Description Language (WS-CDL) [KBR+05], or for interorganizational processes based on ebXML and related standards (cf. [HHK06] for an overview). The integration of composition and choreography languages is currently one of the main research topics in this area [MH05, WHM06].

Today, business process management is an important research area that combines insights from business administration, organization theory, computer science, and computer supported cooperative work. Furthermore, it is a considerable market for software vendors, IT service providers, and business consultants.

2.2 Definition of Business Process Management

Since the beginnings of organization theory, several definitions for business processes have been proposed. Nordsieck in the early 1930s describes a business process as a sequence of activities producing an output. In this context, an activity is the smallest separable unit of work performed by a work subject [Nor34, pp.27-29]. In this tradition, *Becker and Kugeler* [BK03] propose the following definition:

> "A process is a completely closed, timely and logical sequence of activities which are required to work on a process-oriented business object. Such a

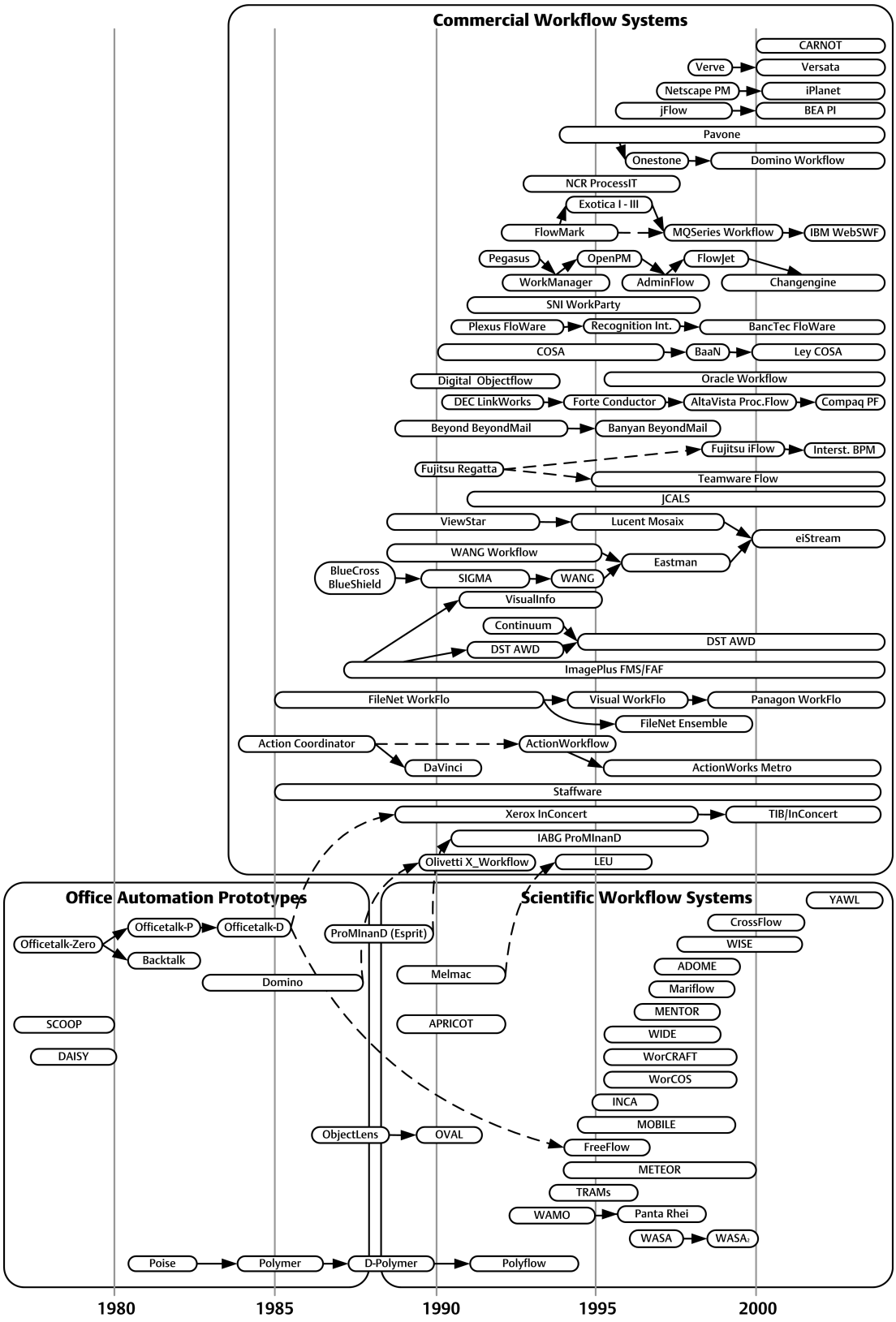


Figure 2.1: History of Office Automation and Workflow Systems [Mue04, p.93]

> process-oriented object can be, for example, an invoice, a purchase order or a specimen. A business process is a special process that is directed by the business objectives of a company and by the business environment. Essential features of a business process are interfaces to the business partners of the company (e.g. customers, suppliers)."

As *Davenport* puts it [Dav93, p.5], a "process is thus a specific ordering of work activities across time and place, with a beginning, an end, and clearly identified inputs and outputs: a structure for action." *Van der Aalst and Van Hee* add that the order of the activities is determined by a set of conditions [AH02, p.4]. In this context, it is important to distinguish between the business process and several individual cases. Consider a business process such as car production. This process produces cars as an output. The production of one individual car that is sold to customer John Smith is a case. Accordingly, each case can be distinguished from other cases, and a business process can be regarded as a class of similar cases [AH02].

Related to business processes and information systems support, several categorization schemes were proposed. As an extension of *Porter*'s value chain model (see [Por85]), *Van der Aalst and Van Hee* distinguish between production, support, and managerial processes [AH02, p.9]. *Production processes* create products and services of a company that are sold to customers. These processes are of paramount importance since they generate income for the company. *Support processes* establish an environment in which the production processes go smoothly. Therefore, they do not only include maintenance activities, but also marketing and finance. *Managerial processes* direct and coordinate production and support processes. They are basically concerned with defining goals, preconditions, and constraints for the other processes. *Leymann and Roller* provide a classification scheme[1] for processes based on their *business value* and their *degree of repetition* [LR00]. They use the term *production process* to refer to those processes that have both a high business value and a high degree of repetition. *Administrative processes* are also highly repetitive, but of little business value. Furthermore, *collaborative processes* are highly valuable, but hardly repeatable. Finally, *ad hoc processes* are neither repetitive

[1] The authors refer to the GIGA group who originally introduced the scheme.

nor valuable. *Leymann and Roller* conclude that information systems support should focus on production processes. In particular, workflow management systems are discussed as a suitable tool. Further classifications can be found, for example, in [DHA05].

Business process management can be defined as the set of all management activities related to business processes. In essence, the management activities related to business processes can be idealistically arranged in a life cycle. Business process management life cycle models have been described for instance in [AH02, Mue04, DHA05]. In the remainder of this section, we mainly follow the life cycle proposed in [Mue04, pp.82-87], firstly, because it does not only include activities but also artifacts, and secondly, because it consolidates the life cycle models for business process management reported in [Hei96, GS95, SD95, NPW03]. It shares the activities analysis, design, and implementation with the general process of information systems development identified by [WW90]. Altogether, the life cycle comprises the management activities of analysis, design, implementation, enactment, monitoring, and evaluation. The solid arcs represent the typical order of these activities, while the dotted arcs depict atypical feedback loops (see Figure 2.2).

Analysis The business process management life cycle is entered with an analysis activity (see Figure 2.2). This analysis covers both the environment of the process and the organization structure. The output of this step is a set of requirements for the business process, such as performance goals.

Design These requirements drive the subsequent design activity. In particular, the design includes the identification of process activities, the definition of their order, the assignment of resources to activities, and the definition of the organization structure. These different aspects of process design are typically formalized as a business process model. This model can be tested in a simulation if it meets the design requirements.[2]

Implementation The process model is then taken as input for the implementation. In this phase, the infrastructure for the business process is set up. This includes

[2]Note that zur Muehlen considers simulation as a separate activity related to evaluation [Mue04, p.86], but this neglects the fact that simulation is always done to evaluate different design alternatives.

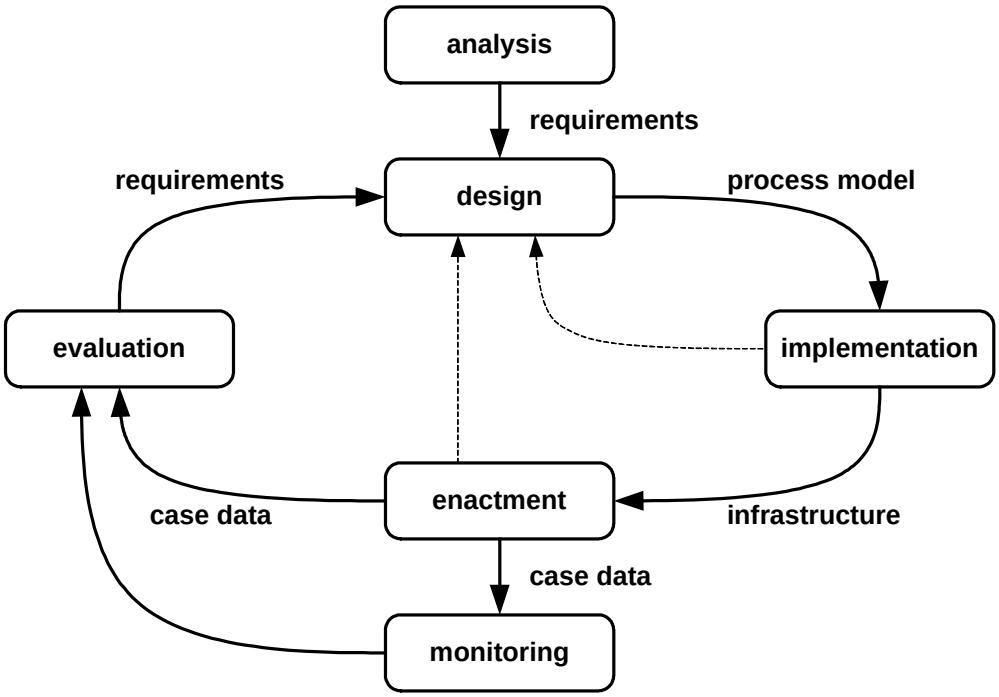


Figure 2.2: Business process management life cycle

(among others) training of staff, provision of a dedicated work infrastructure, or the technical implementation and configuration of software. If the process execution is to be supported by dedicated information systems, the process model is used as a blueprint for the implementation.

Enactment As soon as the implementation is completed, the actual enactment of the process can begin. In this phase, the dedicated infrastructure is used to handle individual cases covered by the business process. The enactment produces information such as consumption of time, resources, materials, etc. for each handled cases. This data can be used as input for two subsequent activities: monitoring and evaluation.

Monitoring is a continuous activity that is performed with respect to each individual case. Depending on process metrics, as, for instance, maximum waiting time for a certain process activity, monitoring triggers respective counteractions if such a metric indicates a problematic situation.

Evaluation on the other hand considers case data on an aggregated level. The perfor-

mance results are compared with the original requirements and sources of further improvement are discussed. In this way, evaluation leads to new requirements that are taken as input in the next turn of the business process management life cycle.

The business process management life cycle reveals that business process models play an important role in the design, implementation, and enactment phase, especially when information systems support the process enactment. Thus, they are valuable resources for continuous process improvement, quality management, knowledge management, ERP system selection, and software implementation [Ros03]. Current market research supports this relevance, since about 90% of the participating companies in a survey conducted or considered business process modeling [Pal07]. In practice, software tools play a decisive role in performing the various management activities in an efficient and effective manner. There are several commercial and academic tools which support different life cycle activities (see e.g. [AH02, Ch.5]). In order to link these tools the Workflow Management Coalition has proposed 5 interfaces in a reference model [Hol94]. In particular, the availability of tools is important to the modeling of business processes in a correct and consistent way.

2.3 Definition of Business Process Modeling

Before defining business process modeling, the term "modeling" has to be discussed in a more general setting. *Nordsieck* has emphasized that "the utilization of symbols enables the model not only to replace or to complement natural language for the representation of complex matters, but to reveal the notion of the subject matter often in a more comprehensive way as with any other form of representation" [Nor32, p.3]. The most protuberant features of a model are brevity, clarity, precision, and its graphic quality [Nor32, p.3]. *Stachowiak* defines a model as the result of a simplifying mapping from reality that serves a specific purpose [Sta73]. According to this perception, there are three important qualities a model should possess. Firstly, there is a mapping that establishes a representation of natural or artificial originals that can be models itself. Secondly, only those attributes of the original that are considered relevant are mapped to the model; the rest

is skipped. Therefore, the model provides an abstraction in terms of a homomorphism in a mathematical sense [Küh06]. Thirdly, the model is used by the modeler in place of the original at a certain point in time and for a certain purpose. This means that a model always involves pragmatics.

A weakness of *Stachowiak*'s concept of a model is that it implies an epistemological position of positivism.[3] This is, for instance, criticized in [SR98], where the authors propose an alternative position based on insights from critical realism and constructivism.[4] This position regards a model as a "result of a construct done by a modeler" [SR98, p.243]. As such, it is heavily influenced by the subjective perception of the modeler. This fact makes modeling a non-deterministic task (cf. [MR]), which requires standards in order to achieve a certain level of inter-subjectivity. The Guidelines of Modeling (GoM) [BRS95, SR98, BRU00] define principles that serve this standardization purpose. They are applicable for either epistemological positions, positivism and constructivism, because both the choice for a certain homomorphism (positivist position) and the perception of the modeler (constructivist position) introduce subjective elements.

Therefore, the Guidelines of Modeling (GoM) [BRS95, SR98] include six particular principles for achieving inter-subjectivity of models. The first three define necessary preconditions for the quality of models, i.e. correctness, relevance, and economic efficiency, and the other three are optional, i.e. clarity, comparability, and systematic design.

Correctness: Firstly, a model has to be syntactically correct. This requirement demands the usage of allowed modeling primitives and combining them according to predefined rules. Secondly, a model must be semantically correct. Therefore, it has to be formally correct and consistent with the (perception of the) real world.

Relevance: This criterion demands that only interesting parts of the universe of discourse are reflected in the model. It is, therefore, related to the notion of completeness as proposed in [BLN86].

[3] Positivism is the philosophical theory that establishes sensual experience as the single object of human knowledge.

[4] In contrast to positivism, constructivism regards all knowledge as constructed. Therefore, there is nothing like objective knowledge or reality.

Economic Efficiency: This guideline introduces a trade-off between benefits and costs of putting the other criteria into practice. For example, semantic correctness might be neglected to a certain extent, in case achieving it is to expensive.

Clarity: This is a highly subjective guideline demanding that the model must be understood by the model user. It is primarily related to layout conventions or the complexity of the model.

Comparability demands consistent utilization of a set of guidelines in a modeling project. Among others, it refers to naming conventions.

Systematic Design: This guideline demands a clear separation between models in different views (e.g. statical aspects and behavioral aspects) and defined mechanisms to integrate them.

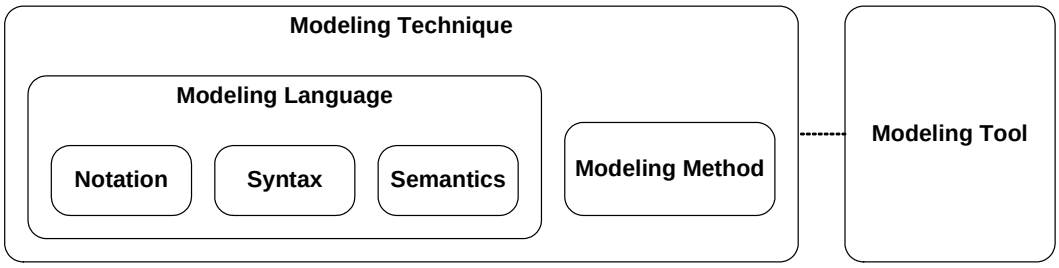


Figure 2.3: Concepts of a modeling technique

Following this line of argument, the explicit definition of a modeling technique appears to be a useful means to address several of these guidelines. A *modeling technique* consists of two interrelated parts: a modeling language and a modeling method[5] (see Figure 2.3). The *modeling language* consists of three parts: syntax, semantics, and optionally, at least one notation. The *syntax* provides a set of constructs and a set of rules how these constructs can be combined. A synonym is modeling grammar [WW90, WW95, WW02]. *Semantics* bind the constructs defined in the syntax to a meaning. This can be done in a mathematical way, e.g. by using formal ontologies or

[5]Several authors use heterogeneous terminology to refer to modeling techniques. Our concept of a modeling language is similar to grammar in [WW90, WW95, WW02] who also use the term method with the same meaning. In [KK02], a modeling method is called "procedure" while the term "method" is used to define a composition of modeling technique plus related algorithms.

operational semantics. The *notation* defines a set of graphical symbols that are utilized for the visualization of models [KK02]. The *modeling method* defines procedures by which a modeling language can be used [WW02]. The result of applying the modeling method is a model that complies with a specific modeling language[6]. Consider, for example, entity-relationship diagrams (ERDs) as defined in [Che76]. Since they define a modeling language and a respective modeling method, ERDs are a modeling technique. Entities and Relationships are syntax elements of its language. They are used to capture certain semantics of a universe of discourse. The notation represents entities as rectangles and relationships as arcs connecting such rectangles carrying a diamond in the middle. Respective procedures, like looking for nouns and verbs in documents, define the modeling method. In practice, *modeling tools* are of crucial importance for the application of a modeling technique. Among others, they support the specification of models, the redundancy controlled administration of models, multi-user collaboration, and model reuse via interfaces to other tools [Ros03]. A recent comparison of business process modeling tools is reported in [AS07].

There are several different approaches to providing a foundation for the correctness and relevance of what is to be put into a model. The following paragraph sketches ontology, speech act theory, and meta-modelling as three alternative foundations. These three approaches are chosen as examples for their wide-spread application in information systems research.

- *Ontology* is the study of being. It seeks to describe what is in the world in terms of entities, categories, and relationships. It is a prominent sub-discipline of philosophy. *Wand and Weber* were among the first to adopt ontology for a foundation of information systems modeling (see e.g. [WW90, WW95]). They make two basic assumptions. Firstly, as information systems reflect what is in the real world they should also be modelled with a language that is capable of representing real-world entities. Secondly, the ontology proposed by *Bunge* [Bun77] is a useful basis for describing the real world. The so-called *Bunge-Wand-Weber* (BWW) Ontology proposed by *Wand and Weber* includes a set of things that can be observed in the world. They should be identified in the process of modeling a specific domain

[6] Instead of model, *Wand and Weber* use the term “script” (cf. [WW90, WW95, WW02]).

and fulfill certain consistency criteria [WW95]. For examples of other ontological models, refer to [WW02, GHW02]. Recently, ontology languages, such as OWL [MH04], have become popular for defining domain ontologies to be used as a component of the semantic web [BLHL01].

- *Speech act theory* is a philosophy of language first proposed by *Austin* [Aus62] and afterwards refined by *Searle* [Sea69]. It emphasizes that language is not only used to make statements about the world that are true or false, but also utilized to do something. A priest, for example, performs a speech act when he pronounces a couple husband and wife. The language action perspective has extended this view after realizing that speech acts do not appear in isolation, but that they are frequently part of a larger conversation [Win88]. *Johannesson* uses this insight to provide a foundation for information systems modeling based on conversations built from speech acts [Joh95]. Coming from the identification of such conversations, *Johannesson* derives consistent structural and behavioral models. Both the foundations in ontology and in speech act theory have in common that they imply two levels of modeling: a general level that is based on abstract entities that the respective theory or philosophy identifies, and a concrete level where the modeler identifies instances of these abstract entities in his modeling domain.

- *Metamodeling* frees modeling from philosophical assumptions by extending the subject of the modeling process to the general (i.e. meta) level. The philosophical theory of this level, such as for instance an ontology, is replaced by a metamodel. The difference to an ontological foundation is that a metamodel does not claim any epistemological validity. Essentially, the metamodel identifies the abstract entities that can be used in the process of designing models, i.e. in other words, the metamodel represents the modeling language (see e.g. [AK01a, KK02, Küh06]). The flexibility gained from this meta-principle comes at the cost of relativism: as a metamodel is meta relative to a model, it is a model itself. Therefore, a metamodel can also be defined for the metamodel and it is called metametamodel. This regression can be continued to infinity without ever reaching an epistemological ground.[7] Most modeling frameworks define three or four modeling levels (see

[7] This negation of a theoretical foundation of a modeling language has some similarities with approaches

UML's Meta Object Facility [OMG02], CASE Data Interchange Format (CDIF) [Fla98], or Graph Exchange Language (GXL) [WKR01]). The definition of a modeling language based on a metamodel is more often used than the explicit reference to a philosophical position. Examples of metamodeling can be found in [ÖG92, Öst95, Fla98, Sch98a, Sch00, AK01b, AK01a, AK03]. Several tools like MetaEdit [SLTM91, KLR96], Protegé [NFM00], or ADONIS [JKSK00] support metamodeling in such a way that modeling languages can be easily defined by the user. For the application of the meta principle in other contexts, refer to [Neu88, Str96].

The meta-hierarchy provides a means to distinguish different kinds of models. Still, a model can never be a metamodel by itself, but only relative to a model for which it defines the modeling language. Models can also be distinguished depending on the mapping mechanism [Str96, p.21]: *Non-linguistic* models capture some real-world aspects as material artifacts or as pictures. *Linguistic* models can be representational, verbal, logistic, or mathematical. Models also serve diverse purposes. Focusing on business administration, *Kosiol* distinguishes descriptive models, explanatory models, and decision models [Kos61]. In this context, descriptive models capture objects of a certain area of discourse and represent them in a structured way. Beyond that, explanatory models define dependency relationships between nomological hypotheses. These serve as general laws to explain real-world phenomena, with a claim for empirical validity. Finally, decision models support the deduction of actions. This involves the availability of a description model to formalize the setting of the decision, a set of goals that constraint the design situation, and a set of decision parameters.

Against this background, the terms *business process model*, *business process modeling language*, and *business process modeling* can be defined as follows:

- A *business process model* is the result of mapping a business process. This business process can be either a real-world business process as perceived by a modeler, or a business process conceptualized by a modeler.

that emphasize that models are not mappings from the real world, but products of negotiations between different stakeholders, as in [HK89, Sim06].

- *Business process modeling* is the human activity of creating a business process model. Business process modeling involves an abstraction from the real-world business process, because it serves a certain modeling purpose. Therefore, only those aspects relevant to the modeling purpose are included in the process model.
- *Business process modeling languages* guide the procedure of business process modeling by offering a predefined set of elements and relationships for the modeling of business processes. A business process modeling language can be specified using a metamodel. In conjunction with a respective method, it establishes a business process modeling technique.

This definition requires some comments. In contrast to [Sta73], it does not claim that the business process model is an abstraction and serves a purpose. These attributions involve some problems about whether a model always has to be abstract or to serve a purpose. Instead, the procedure of business process modeling is characterized in such a way that it is guided by abstraction and a purpose in mind. This is important as a model is not just a "representation of a real-world system" (as *Wand and Weber* put it [WW90, p.123]), but a design artifact, in the sense of *Hevner et al.* [HMPR04], that itself becomes part of the real world as soon as it is created. Beyond that, business process models can be characterized as linguistic models that are mainly representational and mathematical. The representational aspect points to the visual notation of a business process modeling language, while the mathematical notion refers to the formal syntax and semantics. In practice, business process models are often used for documentation purposes [DGR+06]. Therefore, they can be regarded as descriptive models for organization and information systems engineers. Still, they also serve as explanatory and decision models for the people who are involved in the actual processing of cases. In this thesis, the focus is on the descriptive nature of business process models.

2.4 Business Process Modeling and Errors

It is a fundamental insight of software engineering that design errors should be detected as early as possible (see e.g. [Boe81, WW02, Moo05]). The later errors are detected,

the more rework has to be done, and the more design effort has been at least partially useless. This also holds for the consecutive steps of analysis, design, and implementation in the business process management life cycle (cf. e.g. [Ros06, PH07]). In the design phase, process models are typically created with semi-formal business process modeling languages while formal executable models are needed for the implementation. This problem is often referred to as the gap between business process design and implementation phase (see e.g. [MR04]). Therefore, the Guidelines of Process Modeling stress correctness as the most important quality attribute of business process models [BRU00].

In order to provide a better understanding of potential errors in business process models, it is proposed to adapt the information modeling process as identified by *Frederiks and Van der Weide* [FW06]. This process can also serve as a framework for discussing business process modeling in the analysis and design phase of the business process management life cycle. Furthermore, it covers several steps to provide quality assurance in the modeling phase which is of paramount importance for the success of modeling projects (see e.g. [Ros06]). Figure 2.4 gives a business process modeling process mainly inspired by [FW06] and consisting of eight steps. In accordance with *Van Hee et al.* [HSSV06], it is proposed to first verify the process model (step 6) before validating it (step 7-8).

The business process modeling process starts with collecting information objects relevant to the domain (*step 1*). Such information objects include documents, diagrams, pictures, and interview recordings. In *step 2*, these different inputs are verbalized to text that serves as a unifying format. This text is rearranged according to some general guideline of how to express facts (*step 3*) yielding an informal specification. The following step (*step 4*) takes this informal specification as a basis to discover modeling concepts from and to produce a normalized specification. This normal form specification is then mapped to constructs of the process modeling language (*step 5*) in order to create a business process model. These models have to be verified for internal correctness (*step 6*) before they can be paraphrased back to natural language (*step 7*) in order to validate them against the specification (*step 8*). In steps 6-8, the order of activities follows the proposal of *Van Hee et al.* [HSSV06]. It is a good idea to first verify the internal correctness of a model before validating it against the specification, as this prevents incorrect models from being unnecessarily validated.

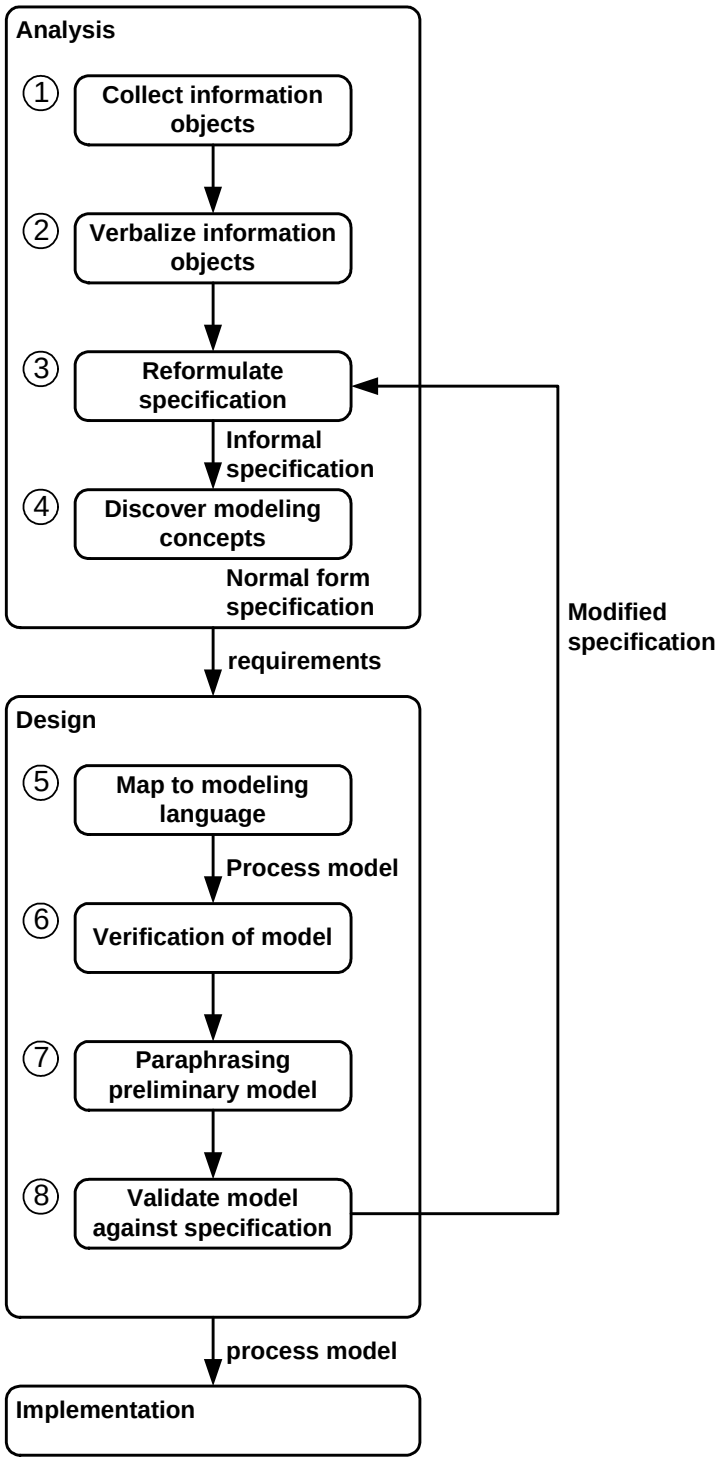


Figure 2.4: Business process modeling process in detail, adapted from [FW06]

The business process modeling process points to two categories of potential errors based on the distinction of verification and validation. This distinction follows the terminology of the Petri nets community (see e.g. *Valmari* [Val98, pp.444]), the conceptual modeling community (see e.g. *Hoppenbrouwers, Proper, and Van der Weide* [HPW05]) and the software engineering community (see e.g. *Boehm* [Boe79], *Sommerville* [Som01]). Different terms for similar concepts are used in *Soffer and Wand* [SW04].

- *Verification* addresses both the general properties of a model and the satisfaction of a given formula by a model. Related to the first aspect, formal correctness criteria play an important role in process modeling. Several criteria have been proposed including soundness for Workflow nets [Aal97], relaxed soundness [DR01], or well-structuredness (see [DZ05] for a comparison). The second aspect is the subject of model checking and involves issues like separation of duty constraints, which can be verified, for example, by using linear temporal logic (LTL), see e.g. [Pnu77].

- Beyond that, *validation* addresses the consistency of the model with the universe of discourse. As it is an external correctness criterion, it is more difficult and more ambiguous to decide. While verification typically relies on an algorithmic analysis of the process model, validation requires the consultation of the specification and discussion with business process stakeholders. SEQUAL can be used as a conceptual framework to validate different quality aspects of a model [LSS94, KSJ06].

In this thesis, we will refer to *formal errors* in connection with the internal correctness of business process models. Formal errors can be identified via verification. Furthermore, we use the term *inconsistencies* to refer to a mismatch of model and specification. Inconsistencies are identified by validation. In a general setting, *error detection* is related to both verification and validation [Val98, p.445]. In the context of this thesis, we focus on *error detection related to verification* and, in particular, to the question which combination of model elements affects the verification of a correctness criterion for a business process model.

Up to now, little empirical work has been conducted on formal errors of business process models in practice. One reason for that is that large repositories of business process models capture specific and valuable business knowledge of industrial or consulting companies. This presents a serious problem for academia since practical modeling experience can hardly be reflected in a scientific way. *Thomas* [Tho05] calls this the "dilemma" of modeling research. One case of a model that is, at least partially, publicly available is the SAP reference model. It has been described in [CKL97, KT98] and is referred to in many research papers (see e.g. [FL03, LS02, MNN05c, RA07, TS06]). The extensive database of this reference model contains almost 10,000 sub-models, 604 of them being non-trivial EPCs [CKL97, KT98]. The verification of these EPC models has shown that there are several formal errors in the models (cf. [ZR96, DAV05, DJV05, MMN$^+$06b]). In [MMN$^+$06b], the authors identify a lower bound for the number of errors of 34 (5.6%), using the relaxed soundness criterion. In another survey, *Gruhn and Laue* [GL07] analyze a collection of 285 EPCs mainly taken from master theses and scientific publications. From these 285 models 30% had trivial errors and another 7% had non-trivial errors. These first contributions highlight that errors are indeed an issue in business process models.

2.5 Summary

In this chapter, we discussed the backgrounds of business process management and defined important terms related to it. Furthermore, we sketched the importance of business process modeling for the business process management life cycle. Since process models are created in the early design phase, they should be free from errors in order to avoid expensive rework and iterations in subsequent phases. In the following chapters, we concentrate on Event-driven Process Chains (EPCs), which are frequently used for business process modeling. Based on a formal semantics definition, we identify verification techniques to detect errors.

Chapter 3

Event-driven Process Chains (EPC)

This chapter provides a comprehensive overview of Event-driven Process Chains (EPCs) and introduces a novel definition of EPC semantics. EPCs became popular as a conceptual business process modeling language in the context of reference modeling. Reference modeling refers to the documentation of generic best practices in a model, for example, for accounting processes in a business domain. Furthermore, it is claimed that reference models can be reused and adapted as best-practice recommendations in individual companies (see [KHHS93, Har94, Kru96, Fra99, Sil01a, Sil01b, Bro03, FL03, RA07, Fet06]). The roots of reference modeling can be traced back to the Kölner Integrationsmodell (KIM) [Gro68, Gro74] that was developed in the 1960s and 1970s. In the 1990s the Institute of Information Systems (IWi) in Saarbrücken worked on a project with SAP to define a suitable business process modeling language to document the processes of the SAP R/3 enterprise resource planning system. There were two results from this joint effort: firstly, the definition of EPCs [KNS92], and secondly, the documentation of the SAP system in the SAP Reference Model (see [CKL97, KT98]). The extensive database of this reference model contains almost 10,000 sub-models, 604 of them non-trivial EPC business

process models. The SAP reference model had a huge impact: on the one hand, several researchers refer to it in their publications (see [ZR96, LS02, FL03, RA07, MNN05c, TS06, Sta06]); on the other hand, it motivated the creation of EPC reference models in further domains including computer integrated manufacturing [Sch87, Sch98b], logistics [Kru96], or retail [BS04]. The wide-spread application of EPCs in business process modeling theory and practice is supported by their coverage in seminal text books for business process management and information systems in general (see e.g. [Sch98a, Sch00, BKR03, STA05, HN05, LLS06]). EPCs are frequently used in practice due to a high user acceptance [SL05] and extensive tool support. Some examples of tools that support EPCs are *ARIS Toolset* of IDS Scheer AG, *AENEIS* of ATOSS Software AG, *ADONIS* of BOC GmbH, *Visio* of Microsoft Corp., *Nautilus* of Gedilan Consulting GmbH, or *Bonapart* by Pikos GmbH. In order to facilitate the interchange of EPC business process models between these tools there is a tool neutral interchange format called EPC Markup Language (EPML) [MN02, MN03b, MN03c, MN04a, MN04c, MN05, MN06].

The remainder of this chapter is structured as follows. Section 3.1 gives a brief informal description of EPC syntax and semantics and introduces the notation by the help of an example. After that, Section 3.2 discusses several approaches to EPC syntax formalization and consolidates them in one definition. Section 3.3 presents various extensions that were proposed for EPCs. Section 3.4 covers different approaches to formal semantics of EPCs and introduces the semantics definition that is used in this thesis. Furthermore, a respective implementation of these semantics in ProM is described. Finally in Section 3.5, EPCs are compared to other business process modeling languages. The chapter concludes with a summary in Section 3.6.

3.1 EPC Notation

The Event-driven Process Chain (EPC) is a business process modeling language for the representation of temporal and logical dependencies of activities in a business process (see [KNS92]). It is utilized in the ARchitecture of Integrated Information Systems (ARIS) by Scheer [Sch98a, Sch00, STA05] as the central method for the conceptual integration of the functional, organizational, data, and output perspective in information

systems design. EPCs offer *function type* elements to capture activities of a process and *event type* elements describing pre-conditions and post-conditions of functions. Some EPC definitions also include *process interface type* elements. A process interface is a syntax element that links two consecutive EPCs: at the bottom of the first EPC, a process interface points to the second EPC, and at the beginning of the second, there is a process interface representing the preceding EPC. Syntactically, it is treated like a function since it represents a subsequent process that can be regarded as a business activity. Furthermore, there are three kinds of *connector types* including AND (symbol $\wedge$), OR (symbol $\vee$), and XOR (symbol $\times$) for the definition of complex routing rules. Connectors have either multiple incoming and one outgoing arc (join connectors) or one incoming and multiple outgoing arcs (split connectors). As a syntax rule, functions and events have to alternate, either directly or indirectly when they are linked via one or more connectors. Furthermore, OR- and XOR-splits after events are not allowed, since events cannot make decisions. Control flow arcs are used to link these elements.

The informal (or intended) semantics of an EPC can be described as follows. The AND-split activates all subsequent branches in concurrency. The XOR-split represents a choice between one of several alternative branches. The OR-split triggers one, two or up to all of multiple branches based on conditions. In both cases of the XOR- and OR-split, the activation conditions are given in events subsequent to the connector. Accordingly, splits from an event to multiple functions are forbidden with XOR and OR as the activation conditions do not become clear in the model. The AND-join waits for all incoming branches to complete, then it propagates control to the subsequent EPC element. The XOR-join merges alternative branches. The OR-join synchronizes all active incoming branches. This feature is called non-locality since the state of all transitive predecessor nodes has to be considered.

Figure 3.1 shows an EPC model for a loan request process as described in *Nüttgens and Rump* [NR02]. The start event *loan is requested* signals the start of the process and the precondition to execute the *record loan request* function. After the post-condition *request is recorded*, the process continues with the function *conduct risk assessment* after the XOR-join connector. The subsequent XOR-split connector indicates a decision. In case of a *negative risk assessment*, the function *check client assessment* is performed.

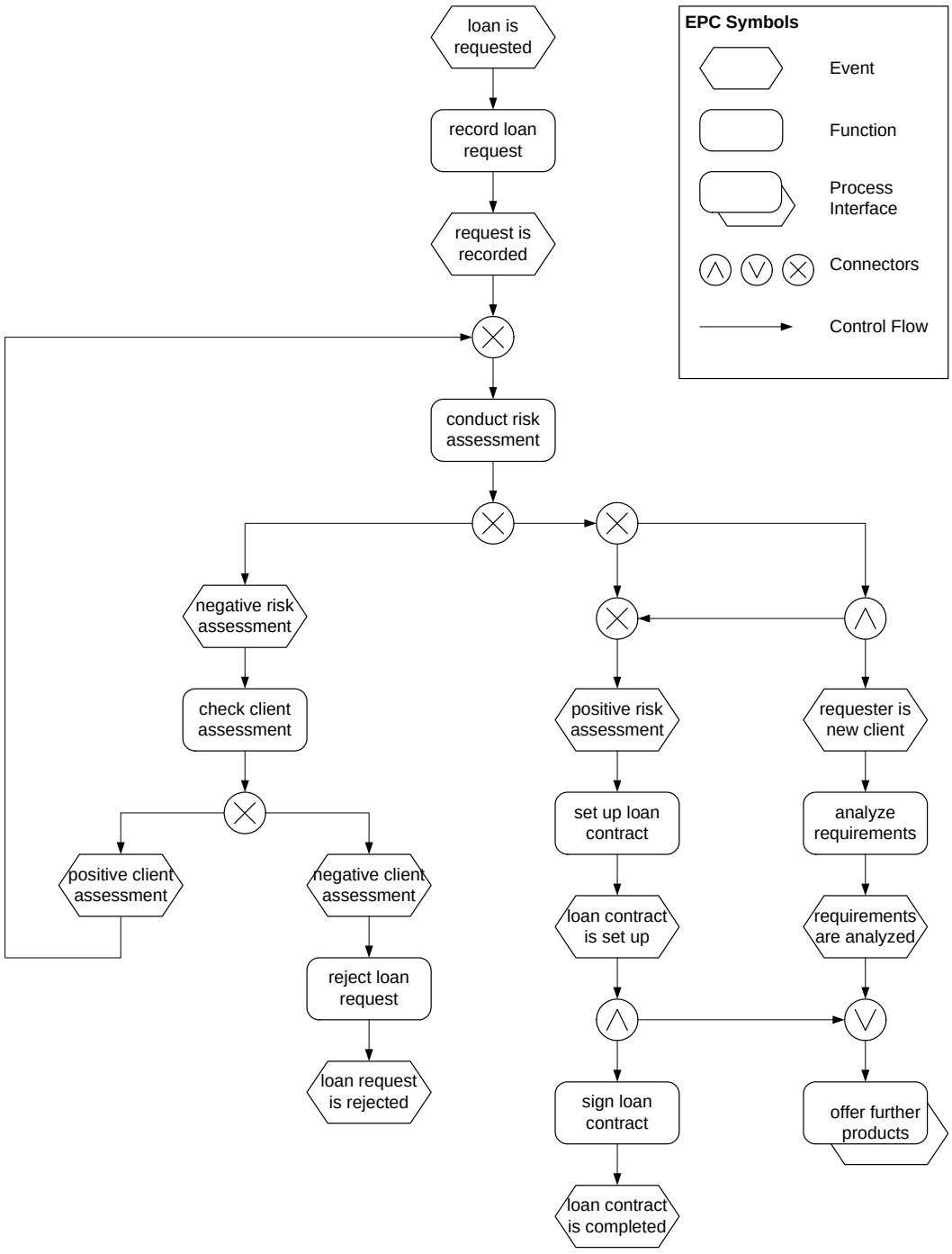


Figure 3.1: EPC for a loan request process [NR02]

The following second XOR-split marks another decision: in case of a *negative client assessment* the process ends with a rejection of the loan request; in case of a *positive client assessment*, the *conduct risk assessment* function is executed a second time under consideration of the positive client assessment. If the risk assessment is not negative, there is another decision point to distinguish new clients and existing clients. In case of an existing client, the *set up loan contract* function is conducted. After that, the AND-split indicates that two activities have to be executed: first, the *sign loan contract* function; and second, the *offer further products* subsequent process (represented by a process interface). If the client is new, the *analyze requirements* function has to be performed in addition to setting up the loan contract. The OR-join waits for both functions to be completed if necessary. If the *analyze requirements* function will not be executed in the process, it continues with the subprocess immediately. The *offer further products* process interface basically triggers a subsequent process (see Figure 3.2) for repeatedly offering products until the offering process is completed.

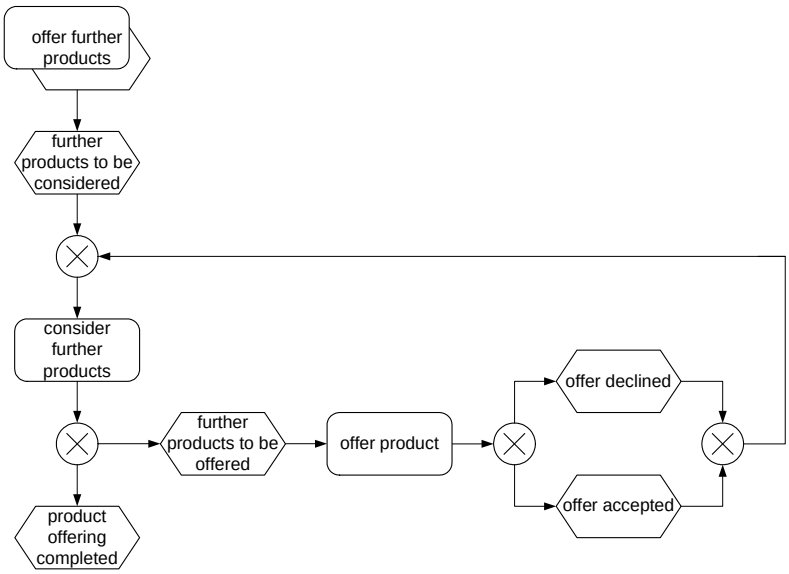


Figure 3.2: EPC for offering further products

3.2 EPC Syntax

There is not just one, but several approaches towards the formalization of EPC syntax. A reason for that is that the original EPC paper introduces them only in an informal way (see [KNS92]). This section gives a historical overview of EPC syntax definitions and joins them into one definition. Please note that we first discuss only standard control flow elements. Extensions are presented in Section 3.3.

3.2.1 Approaches to EPC Syntax Formalization

In *Langner, Schneider, and Wehler* [LSW97a, LSW98], the authors provide a graph-based formalization of EPC syntax distinguishing four types of nodes: function, event, connector, and process interface. Arcs connect elements of these four node types in such a way that the EPC is a simple, directed, and coherent graph. Furthermore, the authors define the following constraints: (a) There are no arcs between elements of the same type, (b) The cardinality of predecessor and successor sets is less or exactly one for events and exactly one for functions and process interfaces, and (c) The border of the EPC graph consists of event type elements only.

Keller and Teufel [KT98, pp.158] provide a formal definition of EPCs in their book on the SAP Reference Model. Beyond the four element types function, event, process interface (there called process sign), and connector, they introduce a concept of model hierarchy depending on hierarchical (or hierarchically ranked) functions. Those hierarchical functions represent a call to a subprocess. Moreover, the authors identify additional constraints for the EPC graph: (a) Connections between connectors must be acyclic, and (b) EPCs have at least three nodes: one start and one end event as well as one function.

The syntax formalization by *Van der Aalst* [Aal99] defines the notion of a path in order to distinguish event-function connectors and function-event connectors. If a connector c is on a path of several consecutive connectors, it is an event-function connector if all paths to it via other connectors start with events, and all paths from it via other connectors lead to functions. Function-event connectors are defined analogously. It is an

additional constraint that each connector is either an event-function or a function-event connector.

In the doctoral thesis of *Rump* [Rum99, pp.79], the EPC syntax definition is separated into two parts, a flat EPC schema and a hierarchical EPC schema. The definition of a flat EPC schema essentially reflects the element types and properties as described above. In addition to that, Rump introduces an initial marking of an EPC. This initial marking must be a subset of the power set of all start events, and each start event must be included in at least one initial marking. A hierarchical EPC schema contains one or more flat EPC schemas and a hierarchy relation that maps hierarchical functions and process interfaces to EPCs. The hierarchy relation must establish an acyclic graph. A similar syntax definition is presented in [NR02].

The alternation between events and functions with several connectors in between was first enforced by the definition of *Van der Aalst*. Yet, all paths between the elements have to be considered. Therefore, *Mendling and Nüttgens* provide a syntax definition based on two arc types: event-function arcs and function-event arcs [MN03a]. As a constraint, event-function arcs must have either events or event-function connectors as source nodes, and functions or event-function connectors as target nodes. In analogy, a similar constraint must hold for function-event arcs. An advantage of this definition is that EPC syntax validation does not require path expansion for each connector in a chain.

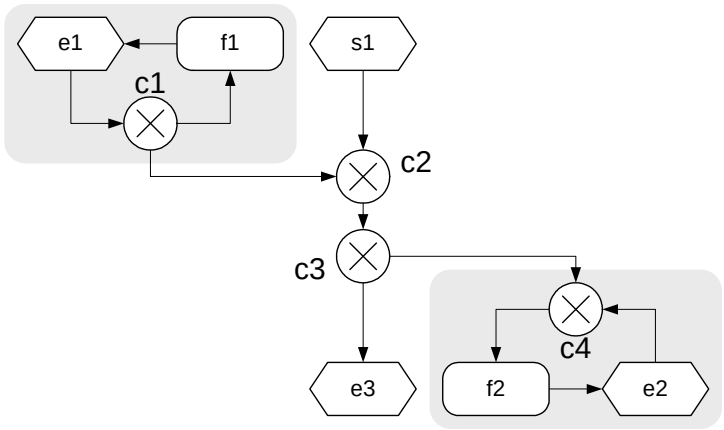


Figure 3.3: An EPC with nodes that have no path from a start event and that have no path to an end event

While the different syntax formalizations cover an extensive set of properties, there is one syntax problem which is not addressed. Figure 3.3 shows an EPC that has two undesirable properties. Firstly, for the nodes $e1$, $f1$, and $c1$ there is no path from a start node to reach them. Secondly, there is no path from the nodes $e2$, $f2$, and $c4$ to an end node. Since such a structure is not meaningful, we will add a new syntax requirement that each node of an EPC must be on a path from a start to an end node.

Table 3.1 summarizes the presented approaches towards EPC syntax formalization. Further work related to EPC syntax is listed in [NR02]. The following section provides a formal EPC syntax definition that consolidates the different approaches.

Table 3.1: Overview of approaches to EPC syntax formalization

	[LSW97a]	[KT98]	[Aal99]	[Rum99]	[MN03a]
Cardinality Constraints	+	+	+	+	+
Fnct.-Event Alternation	+/-	+/-	+	+	+
No Connector cycles	-	+	-	+	+
Hierarchy	-	+	-	+	+
No Hierarchy cycles	-	-	-	+	+
Initial Marking	-	-	-	+	+
Nodes on start-end-path	-	-	-	-	-

3.2.2 Formal Syntax Definition of Flat EPCs

The subsequent syntax definition of flat EPCs essentially follows the presentation in [NR02] and [MN03a]. If it is clear from the context that a flat EPC is discussed, the term EPC will be used instead for brevity. Please note that an initial marking as proposed in [Rum99, NR02] is not included in the syntax definition, but discussed in the context of EPC semantics in Section 3.4.4.

Definition 3.1 (Flat EPC)**.** A flat $EPC = (E, F, P, C, l, A)$ consists of four pairwise disjoint and finite sets E, F, C, P, a mapping $l : C \rightarrow \{and, or, xor\}$, and a binary relation $A \subseteq (E \cup F \cup P \cup C) \times (E \cup F \cup P \cup C)$ such that

– An element of E is called *event*. $E \neq \emptyset$.

- An element of F is called *function*. $F \neq \emptyset$.
- An element of P is called *process interface*.
- An element of C is called *connector*.
- The mapping l specifies the type of a connector $c \in C$ as and, or, or xor.
- A defines the control flow as a coherent, directed graph. An element of A is called an *arc*. An element of the union $N = E \cup F \cup P \cup C$ is called a *node*.

In order to allow for a more concise characterization of EPCs, notations are introduced for preset and postset nodes, incoming and outgoing arcs, paths, transitive closure, corona, and several subsets.

Definition 3.2 (Preset and Postset of Nodes)**.** Let N be a set of $nodes$ and $A \subseteq N \times N$ a binary relation over N defining the arcs. For each $node\ n \in N$, we define $\bullet n = \{x \in N | (x, n) \in A\}$ as its *preset*, and $n\bullet = \{x \in N | (n, x) \in A\}$ as its *postset*.

Definition 3.3 (Incoming and Outgoing Arcs)**.** Let N be a set of $nodes$ and $A \subseteq N \times N$ a binary relation over N defining the arcs. For each $node\ n \in N$, we define the set of incoming arcs $n_{in} = \{(x, n) | x \in N \wedge (x, n) \in A\}$, and the set of outgoing arcs $n_{out} = \{(n, y) | y \in N \wedge (n, y) \in A\}$.

Definition 3.4 (Paths and Connector Chains)**.** Let $EPC = (E, F, P, C, l, A)$ be a flat EPC and $a, b \in N$ be two of its nodes. A *path* $a \hookrightarrow b$ refers to the existence of a sequence of EPC nodes $n_1, \ldots, n_k \in N$ with $a = n_1$ and $b = n_k$ such that for all $i \in 1, \ldots, k$ holds: $(n_1, n_2), \ldots, (n_i, n_{i+1}), \ldots, (n_{k-1}, n_k) \in A$. This includes the empty path of length zero, i.e., for any node $a : a \hookrightarrow a$. If $a \neq b \in N$ and $n_2, \ldots, n_{k-1} \in C$, the path $a \overset{c}{\hookrightarrow} b$ is called *connector chain*. This includes the empty connector chain, i.e., $a \overset{c}{\hookrightarrow} b$ if $(a, b) \in A$. The *transitive closure* A^* contains (n_1, n_2) if there is a non-empty path from n_1 to n_2, i.e., there is a a non-empty set of arcs of A leading from n_1 to n_2. For each $node\ n \in N$, we define its *transitive preset* $*n = \{x \in N | (x, n) \in A^*\}$, and its *transitive postset* $n* = \{x \in N | (n, x) \in A^*\}$.

Definition 3.5 (Upper Corona, Lower Corona)**.** Let $EPC = (E, F, P, C, l, A)$ be a flat EPC and N its set of $nodes$. Then its upper corona is defined as $\overset{c}{*}\mathrm{n} = \{v \in (E \cup F \cup P) | v \overset{c}{\hookrightarrow} n\}$ for some node $n \in N$. It includes those non-connector nodes of the

transitive preset that reach n via a connector chain. In analogy, its lower corona is defined as $n\overset{c}{*}= \{w \in (E \cup F \cup P) | n \overset{c}{\hookrightarrow} w\}$.

Definition 3.6 (Subsets)**.** For an $EPC = (E, F, P, C, l, A)$, we define the following subsets of its nodes and arcs:

- $E_s = \{e \in E \mid |\bullet e| = 0 \wedge |e\bullet| = 1\}$ being the set of start events,
  $E_{int} = \{e \in E \mid |\bullet e| = 1 \wedge |e\bullet| = 1\}$ being the set of intermediate events, and
  $E_e = \{e \in E \mid |\bullet e| = 1| \wedge e\bullet| = 0\}$ being the set of end events.
- $P_s = \{p \in P \mid |\bullet p| = 0 \wedge |p\bullet| = 1\}$ being the set of start process interfaces,
  $P_e = \{p \in P \mid |\bullet p| = 1 \wedge |p\bullet| = 0\}$ being the set of end process interfaces.
- $J = \{c \in C \mid |\bullet c| > 1$ and $|c\bullet| = 1\}$ as the set of join- and
  $S = \{c \in C \mid |\bullet c| = 1$ and $|c\bullet| > 1\}$ as the set of split-connectors.
- $C_{and} = \{c \in C \mid l(c) = and\}$ being the set of and-connectors,
  $C_{xor} = \{c \in C \mid l(c) = xor\}$ being the set of xor-connectors, and
  $C_{or} = \{c \in C \mid l(c) = or\}$ being the set of or-connectors.
- $J_{and} = \{c \in J \mid l(c) = and\}$ being the set of and-join connectors,
  $J_{xor} = \{c \in J \mid l(c) = xor\}$ being the set of xor-join connectors,
  $J_{or} = \{c \in J \mid l(c) = or\}$ being the set of or-join connectors,
- $S_{and} = \{c \in S \mid l(c) = and\}$ being the set of and-split connectors,
  $S_{xor} = \{c \in S \mid l(c) = xor\}$ being the set of xor-split connectors, and
  $S_{or} = \{c \in S \mid l(c) = or\}$ being the set of or-split connectors.
- $C_{EF} = \{c \in C \mid \overset{c}{*}c \subseteq E \wedge c\overset{c}{*} \subseteq (F \cup P)\}$ as the set of event-function connectors (ef-connectors) and
  $C_{FE} = \{c \in C \mid \overset{c}{*}c \subseteq (F \cup P) \wedge c\overset{c}{*} \subseteq E\}$ as the set of function-event connectors (fe-connectors).
- $A_{EF} = A \cap ((E \cup C_{EF}) \times (F \cup P \cup C_{EF}))$ as the set of event-function arcs,
  $A_{FE} = A \cap ((F \cup P \cup C_{FE}) \times (E \cup C_{FE}))$ as the set of function-event arcs.
- $A_s = \{(x, y) \in A \mid x \in E_s\}$ as the set of start-arcs,
  $A_{int} = \{(x, y) \in A \mid x \notin E_s \wedge y \notin E_e\}$ as the set of intermediate-arcs, and
  $A_e = \{(x, y) \in A \mid y \in E_e\}$ as the set of end-arcs.

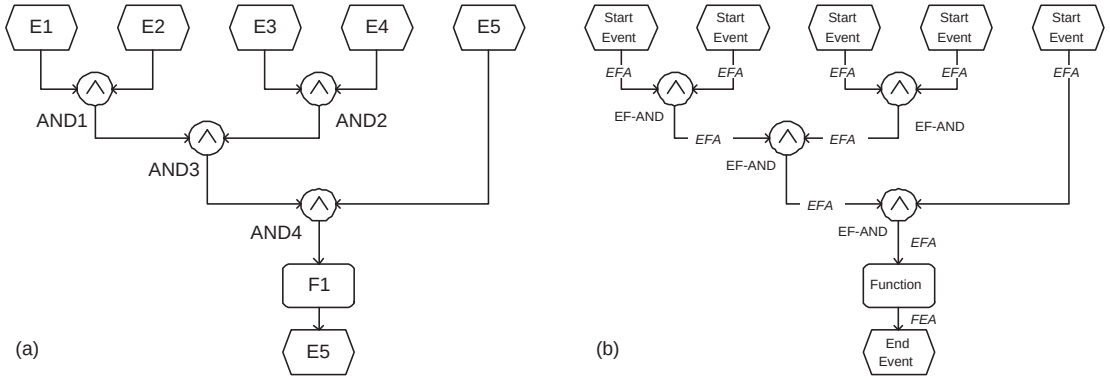


Figure 3.4: An EPC (a) with labelled nodes and (b) its nodes related to the subsets of Definition 3.6.

Figure 3.4 illustrates the different subsets of an EPC. Consider, for example, the connector $AND3$. It is an event-function connector (labelled as *EF-AND*) since its upper corona, i.e. those non-connector nodes from which there is a connector chain to $AND3$, contains only events and its lower corona contains functions only, in this case exactly one. Furthermore, the arc from $AND1$ to $AND3$ is an event-function arc (labelled as *EFA*) since it connects two event-function connectors. Note that arcs from events to event-function connectors and arcs from event-function connectors to functions are event-function arcs, too.

We summarize the EPC syntax requirements as follows.

Definition 3.7 (Syntactically Correct EPC)**.** An $EPC = (E, F, P, C, l, A)$ is called syntactically correct, if it fulfills the requirements:

1. EPC is a simple, directed, coherent, and antisymmetric graph such that $\forall n \in N$: $\exists e_1 \in E_s, e_2 \in E_e$ such that $e_1 \hookrightarrow n \hookrightarrow e_e$.
2. There are no connector cycles, i.e. $\forall a, b \in C$: if $a \neq b$ and $a \overset{c}{\hookrightarrow} b$, then $\nexists b \overset{c}{\hookrightarrow} a$.
3. $|E_s \cup P_s| \geq 1 \wedge |E_e \cup P_e| \geq 1$. There is at least one start node and one end node in an EPC.
4. $|F| \geq 1$. There is at least one function in an EPC.
5. Events have at most one incoming and one outgoing arc.

$\forall e \in E : |\bullet e| \leq 1 \wedge |e\bullet| \leq 1$.
This implies that E_s,E_{int}, and E_e partition E.

6. Functions have exactly one incoming and one outgoing arc.
   $\forall f \in F : |\bullet f| = 1 \wedge |f\bullet| = 1$.
7. Process interfaces have one incoming or one outgoing arc, but not both.
   $\forall p \in P : (|\bullet p| = 1 \wedge |p\bullet| = 0) \vee (|\bullet p| = 0 \wedge |p\bullet| = 1)$.
   This implies that P_s and P_e partition P.
8. Connectors have one incoming and multiple outgoing arcs or multiple incoming and one outgoing arc. $\forall c \in C : (|\bullet c| = 1 \wedge |c\bullet| > 1) \vee (|\bullet c| > 1 \wedge |c\bullet| = 1)$.
   This implies that J and S partition C.
9. Events must have function, process interface, or fe-connector nodes in the preset, and function, process interface, or ef-connector nodes in the postset.
   $\forall e \in E : \bullet e \subseteq (F \cup P \cup C_{FE}) \wedge e\bullet \subseteq (F \cup P \cup C_{EF})$.
10. Functions must have events or ef-connectors in the preset and events or fe-connectors in the postset.
    $\forall f \in F : \bullet f \subseteq (E \cup C_{EF}) \wedge f\bullet \subseteq (E \cup C_{FE})$.
11. Process interfaces are connected to events only.
    $\forall p \in P : \bullet p \subseteq E \wedge p\bullet \subseteq E$.
12. Connectors must have either functions, process interfaces, or fe-connectors in the preset and events or fe-connectors in the postset; or events or ef-connectors in the preset and functions, process interfaces, or ef-connectors in the postset.
    $\forall c \in C : (\bullet c \subseteq (F \cup P \cup C_{FE})) \wedge c\bullet \subseteq (E \cup C_{FE}) \vee$
    $(\bullet c \subseteq (E \cup C_{EF}) \wedge c\bullet \subseteq (F \cup P \cup C_{EF}))$.
13. Arcs either connect events and ef-connectors with functions, process interfaces, and ef-connectors or functions, process interfaces, and fe-connectors with events and fe-connectors.
    $\forall a \in A : (a \in (F \cup P \cup C_{FE}) \times (E \cup C_{FE})) \vee (a \in (E \cup C_{EF}) \times (F \cup P \cup C_{EF}))$.

Given this definition, the EPCs of Figures 3.1 and 3.4 are syntactically correct. In Section 4.3, we define reduction rules for EPCs with relaxed syntax requirements. Relaxed syntactical correctness drops the requirements that the EPC graph is simple and antisymmetric (1), that there are no connector cycles (2), that the set of functions is not

empty (4), and that functions and events have to alternate (9 to 13).

Definition 3.8 (Relaxed Syntactically Correct EPC)**.** An $EPC = (E, F, P, C, l, A)$ is called relaxed syntactically correct if it fulfills the following requirements:

1. EPC is a directed and coherent graph such that $\forall n \in N : \exists e_1 \in E_s, e_2 \in E_e$ such that $e_1 \hookrightarrow n \hookrightarrow e_2$.
2. $|E_s \cup P_s| \geq 1 \wedge |E_e \cup P_e| \geq 1$. There is at least one start node and one end node in an EPC.
3. Events have at most one incoming and one outgoing arc.
   $\forall e \in E : |\bullet e| \leq 1 \wedge |e\bullet| \leq 1$.
4. Functions have exactly one incoming and one outgoing arcs.
   $\forall f \in F : |\bullet f| = 1 \wedge |f\bullet| = 1$.
5. Process interfaces have one incoming or one outgoing arcs.
   $\forall p \in P : (|\bullet p| = 1 \wedge |p\bullet| = 0) \vee (|\bullet p| = 0 \wedge |p\bullet| = 1)$.
6. Connectors have at least one incoming and one outgoing arc such that
   $\forall c \in C : (|\bullet c| = 1 \wedge |c\bullet| \geq 1) \vee (|\bullet c| \geq 1 \wedge |c\bullet| = 1)$.

If an EPC is syntactically correct, then it is also relaxed syntactically correct.

3.2.3 Formal Syntax Definition of Hierarchical EPCs

Hierarchical decomposition is a general principle of many system analysis techniques such as data-flow diagrams, object-oriented analysis, or organization charts (see [Bal98, pp.557] or [Som01, Ch.7]). Hierarchical refinement is also an appropriate technique for the description of complex processes at different levels of granularity [AH02, p.34]. Such deposition techniques were also defined for EPCs (see [NR02, MN03a, IDS03a]). Figure 3.5 gives the example of a return deliveries process that is included in the procurement module of the SAP Reference Model. Within this EPC the *Warehouse* function is hierarchically decomposed to another EPC that is depicted on the right-hand side of the figure. The semantics of such a decomposition is that the subprocess is started when the hierarchical function is activated. When the subprocess is completed, control is forwarded to

the event subsequent to the hierarchical function. In the following, we define hierarchical EPCs in a slightly different way compared to [NR02, MN03a], in order to achieve a clear separation of the EPC and the hierarchy concept. Still, not all requirements of [NR02] are met by the example in Figure 3.5, since only the start event of the subprocess matches the pre-event in the parent process, but not the end events.

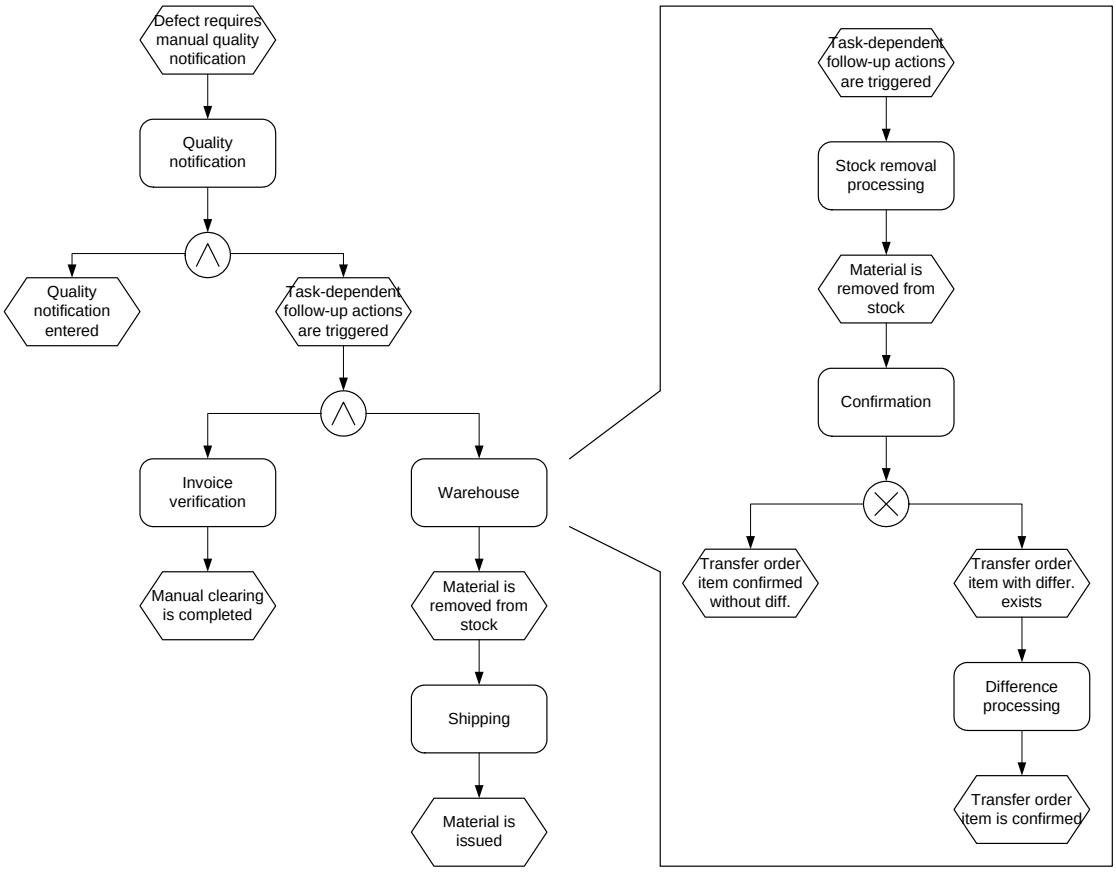


Figure 3.5: The return deliveries process from the SAP Reference Model with a hierarchical decomposition of the Warehouse function.

Definition 3.9 (Hierarchical EPC)**.** A hierarchical EPC $EPC_H = (Z, h)$ consists of a set of EPCs Z and a partial function $h : D \rightharpoonup Z$ on a domain D of decomposed functions and process interfaces such that

– Z is a set of EPCs. $N(z)$ refers to the nodes of one individual EPC $z \in Z$. Accordingly, $E(z), E_e(z), E_s(z), F(z), P(z), C(z)$, and $A(z)$ refer to the sets of events,

start events, end events, functions, process interfaces, connectors, and arcs of an EPC $z \in Z$. We refer to the union of all functions and process interfaces by $F = \bigcup_{z \in Z} F(z)$ and $P = \bigcup_{z \in Z} P(z)$.

- The domain D is a subset of functions and process interfaces of EPCs contained in Z, i.e., $D \subseteq F \cup P$.
- The mapping h specifies a partial function from the domain D to the set of EPCs Z. For a node $d \in D$ such that $h(d) = z$, z is called "subprocess of d" or "process referenced from d".
- $G \subseteq (Z \times Z)$ defines the hierarchy graph for a hierarchical EPC.
  A pair $(z_1, z_2) \in G$ if and only if $\exists d \in (D \cap (F(z_1) \cup P(z_1))) : h(d) = z_2$.

According to [NR02], a syntactically correct hierarchical EPC must fulfill the following constraints.

Definition 3.10 (Syntactically Correct Hierarchical EPC)**.** A hierarchical $EPC_H = (Z, h)$ with the domain D of h is called syntactically correct if it fulfills the following constraints:

1. All EPCs of Z must be syntactical correct flat EPCs.
2. All functions of the domain D map to an EPC of Z.
   $\forall f \in F : f \in D \Rightarrow h(f) \in Z$.
3. All process interfaces map to an EPC of Z.
   $\forall p \in P : p \in D \wedge h(p) \in Z$.
4. If $f \in D$, then the upper corona of f is equal to the set of start events of $h(f)$, i.e., $\overset{c}{*}f = E_s(h(f))$.
5. If $f \in D$, then the lower corona of f is equal to the set of end events of $h(f)$, i.e., $f\overset{c}{*}= E_e(h(f))$.
6. For all $p \in P$ the preset event of p is a subset of the start events of $h(p)$, i.e., $\bullet p \subseteq E_s(h(f))$.
7. For all $p \in P$ the postset event of p is a subset of the end events of $h(p)$, i.e., $p\bullet \subseteq E_e(h(f))$.
8. The hierarchy graph G of EPC_H is acyclic.

While constraint 4 is fulfilled in Figure 3.5, i.e., *task-dependent follow-up actions are triggered* precedes the hierarchical function *warehouse* and it is the start event of the subprocess, constraint 5 is not fulfilled since the post-event to *warehouse* differs from the end events of the subprocess. Yet, if the constraints are met, the hierarchy relation can be used to flatten the hierarchical EPC. Therefore, the event after the *warehouse* function should be renamed to *Transfer order item confirmed with or without difference*. In this case, the relationships between the corona of functions or process interfaces and the start and end events of the referenced EPC can be utilized to merge the EPC with its subprocess as defined in [GRSS05, MS06].

3.2.4 Formal Syntax Definition of Standard EPCs

Throughout the remainder of this thesis, we will have a specific focus on a subset of EPCs that we refer to as standard EPCs.

Definition 3.11 (Standard EPC)**.** A flat $EPC = (E, F, P, C, l, A)$ that has an empty set $P = \emptyset$ is called standard EPC. For brevity P can be omitted in the definition. Accordingly, $EPC = (E, F, C, l, A)$ refers to a standard EPC.

In the following sections and chapters, we will use the terms EPC and standard EPC as synonyms. Furthermore, we assume EPCs to be relaxed syntactically correct.

3.3 EPC Syntax Extensions

Several variants and extensions were proposed for EPCs. Some of them are listed in [SDL05, p.106]. EPC Variants include Real-Time EPC (rEPC) [HWS93], EPC* for workflow execution [ZR96], Object-oriented EPC (oEPC) [SNZ97], Modified EPC (modEPC) [Rit99], Agent-oriented EPC (xEPC) [KHP+00], Yet Another EPC (yEPC) [MNN05a, MNN05c, MNN05d], Nautilus EPCs [KUL06], and Semantic EPCs [TF06a, TF06b]. There is also a plethora of EPCs with non control flow elements (see [Sch98a, Sch00, STA05, Ros95, Sch98c, LA98, Sch02, Kug02, BO02, ST03, BAN03]),

that serve as structured annotations to the process model. In the following subsections, we give an overview of EPC extensions for control flow and configurability (see Table 3.2).

Table 3.2: Overview of EPC extensions for control flow and configurability

Category	Name of Concept	Authors
Control Flow	SEQ-Connector	Priemer [Pri95]
	ET-Connector	Rosemann [Ros95]
	OR_1-Connector	Rosemann [Ros95]
	OR-Connector variants	Rittgen [Rit99, Rit00]
	Fuzzy-Connector	Thomas et al. [THA02, ATM03]
	Multiple-Instance-Connector	Rodenhagen [Rod02]
	Empty-Connector	Mendling et al. [MNN05a]
	Multiple Instances	Mendling et al. [MNN05a]
	Cancellation	Mendling et al. [MNN05a]
Configurability	Configurable Function	Rosemann et al. [RA07]
	Configurable Connector	Rosemann et al. [RA07]
	Configuration Requirement	Rosemann et al. [RA07]
	Configuration Guideline	Rosemann et al. [RA07]
	Configuration Order	Rosemann et al. [RA07]

3.3.1 Control Flow Extensions

Control flow extensions are defined either to introduce expressive power or to provide a clarification of semantics. The *SEQ*-connector is introduced in *Priemer* [Pri95]. It can be used to specify non-parallel, but arbitrary order of activities. As such, a *SEQ* split-join pair captures the semantics of workflow pattern 17 (interleaved parallel routing) as described in [AHKB03]. Furthermore, *Rosemann* introduces an *ET*-connector that explicitly models a decision table and a so-called OR_1 connector to mark branches that are always executed [Ros95]. The motivation of both these proposals is to offer a straight forward way to model certain behavior. In contrast to that, the works of *Rittgen* are motivated by semantic ambiguities of the OR-join (see [Rit99, Rit00]). We will discuss his proposal for *every-time*, *first-come*, and *wait-for-all* OR-joins in Section 3.4.3. The aim of *Thomas et al.* [THA02, ATM03, TD06a] is to provide modeling support for decisions that are taken based on fuzzy information. The authors introduce a *fuzzy XOR*-connector

that takes multiple inputs and triggers alternative outputs. *Rodenhagen* presents *multiple instantiation* as a missing feature of EPCs [Rod02]. He proposes dedicated begin and end symbols to model that a branch of a process may be executed multiple times. Yet, this notation does not enforce that a begin symbol is followed by a matching end symbol.

The work by *Mendling, Neumann, and Nüttgens* on yEPCs [MNN05a, MNN05c, MNN05d] is inspired by missing support for some of the workflow patterns identified in [AHKB03]. In order to capture the semantics of unsupported patterns, three new elements are introduced: empty connector, cancellation area, multiple instantiation. The *empty split* can be interpreted as a hyperarc, for instance, from the event before the empty split to the functions subsequent to it; the *empty join* analogously as a hyperarc from multiple functions before it to its subsequent event. In this context, the split semantics match the deferred choice pattern, and the join semantics the multiple merge pattern. Also interleaved parallel routing and milestones can be represented by the help of empty connectors. *Multiple instantiation* in yEPCs is described similarly as in YAWL by giving the *min* and *max* cardinality of instances that may be created. The *required* parameter specifies an integer number of instances that must finish in order to complete the function. The *creation* parameter specifies whether further instances may be created at run-time (dynamic) or not (static). *Cancellation* areas (symbolized by a lariat) include a set of functions and events. The end of the lariat is connected to a function. When this function is completed, all functions and events in the lariat are cancelled. In [MMN06a] the authors provide a transformation from yEPCs to YAWL.

3.3.2 Configurability Extensions

A first approach towards the configurability of EPCs can be found in *Rosemann* [Ros95, p.245], who defines type operators to describe process alternatives. Configurable EPCs (C-EPCs) as an heir to this work extend EPCs for specification of variation points, configuration constraints, and configuration guidance in a reference model. They play a central role in the realization of an integrated, model-driven Enterprise Systems Configuration life cycle as proposed in [RMRA06].

In a C-EPC, functions and connectors can be configured. For a *configurable function*,

a decision has to be made: whether to perform it in every process instance during run time (ON), or whether to exclude it permanently (OFF), i.e. it will not be executed in any process instance, or whether to defer this decision to run time (OPT), i.e. for each process instance, it has to be decided whether to execute the function or not. *Configurable connectors* subsume build-time connector types that are less or equally expressive. Hence, a configurable connector can only be configured to a connector type that restricts its behavior. A configurable OR-connector may be mapped to a regular OR-, XOR-, AND-connector, or to a single sequence of events and functions (indicated by SEQ_n for some process path starting with node n). A configurable AND-connector may only be mapped to a regular AND-connector. A configurable XOR-connector may be mapped to a regular XOR-connector or to a single sequence SEQ_n. Interdependencies between configurable EPC nodes can be specified via *configuration requirements*, i.e. logical expressions that define constraints for inter-related configuration nodes. *Configuration guidelines* formalize recommendations and best practices (also in the form of logical expressions) in order to support the configuration process semantically. Additional work formalizes C-EPC syntax [RA07], its mapping to EPCs [MRRA06], and its identification from existing systems [JVAR06].

3.4 EPC Semantics

In addition to related work on the syntax of EPCs, there are several contributions towards the formalization of EPC semantics. This section first illustrates the semantical problems related to the OR-join. Then it gives a historical overview of semantical definitions, and provides a formalization for EPCs, that is used in this thesis. Furthermore, we present an implementation of these semantics as a ProM plug-in that generates the reachability graph for a given EPC.

3.4.1 Informal Semantics as a Starting Point

Before discussing EPC formalization problems, we need to establish an informal understanding of state representation and state changes of an EPCs. Although we provide

a formal definition not before Section 3.4.5, the informal declaration of state concepts helps to discuss formalization issues in this section. The *state*, or *marking*, of an EPC is defined by assigning a number of *tokens* (or process folders) to its arcs.[1] The formal semantics of an EPC define which state changes are possible for a given marking. These state changes are formalized by a *transition relation*. A node is called *enabled* if there are enough tokens on its incoming arcs that it can fire, i.e. a state change defined by a transition can be applied. This process is also called *firing*. A firing of a node n consumes tokens from its input arcs n_{in} and produces tokens at its output arcs n_{out}. The formalization of whether an OR-join is enabled is a non-trivial issue since not only the incoming arcs have to be considered. The sequence $\tau = n_1 n_2 ... n_m$ is called a firing sequence if it is possible to execute a sequence of steps, i.e. after firing n_1 it is possible to fire n_2, etc. Through a sequence of firings, the EPC moves from one *reachable* state to the next. The *reachability graph* of an EPC represents how states can be reached from other states. A marking that is not a final marking, but from which no other marking can be reached, is called a *deadlock*. The notion of an initial and a final marking will be formally defined in Section 3.4.5.

3.4.2 EPC Formalization Problems

We have briefly stated that the OR-join synchronizes all active incoming branches. This bears a non-trivial problem: if there is a token on one incoming arc, does the OR-join have to wait or not? Following the informal semantics of EPCs, it is only allowed to fire if it is not possible for a token to arrive on the other incoming arcs (see [NR02]). In the following subsection, we will show what the formal implications of these intended semantics are. Before that, we present some example EPCs, the discussion of which raises some questions that will not be answered immediately. Instead, we revisit them later on to illustrate the characteristics of different formalization approaches.

Figure 3.6(a) shows an EPC with an OR-join on a loop. There is a token on arc $a2$

[1] This state representation based on arcs reflects the formalization of *Kindler* [Kin03, Kin04, Kin06] and can be related to arcs between tasks in YAWL that are interpreted as implicit conditions [AH05]. Other approaches assign tokens to the nodes of an EPC, e.g., [Rum99]. Later, we will make a distinction between state and marking in our formalization of EPC operational semantics.

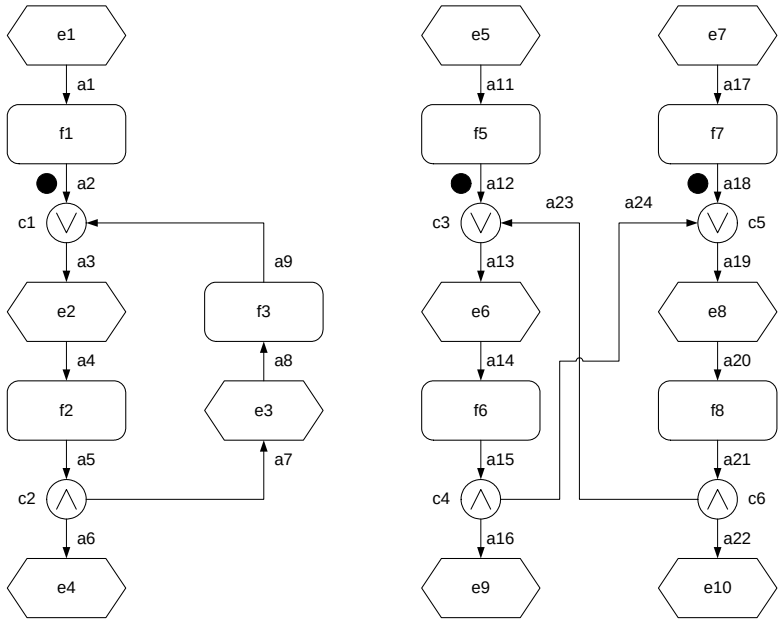


Figure 3.6: EPCs (a) with one OR-join and (b) with two OR-joins on the loop

from function $f1$ to the OR-join $c1$. The question is whether $c1$ can fire. If it could fire, then it would be possible for a token to arrive on arc $a9$ from $f3$ to the join. This would imply that it should wait and not fire. On the other hand, if it must wait, it is not possible that a token might arrive at $a9$. Figure 3.6(b) depicts an EPC with two OR-joins, $c3$ and $c5$, on a loop which are both enabled (cf. [ADK02]). Here, the question is whether both or none of them can fire. Since the situation is symmetrical, it seems unreasonable that only one of them should be allowed to fire.

The situation might be even more complicated, as Figure 3.7 illustrates (cf. [Kin03, Kin04, Kin06]). This EPC includes a loop with three OR-joins: $c1$, $c3$, and $c5$, all of which are enabled. Following the informal semantics, the first OR-join $c1$ is allowed to fire if it is not possible for a token to arrive on arc $a21$ from the AND-split $c6$. To put it differently, if $c1$ is allowed to fire, it is possible for a token to arrive on arc $a7$ that leads to the OR-join $c3$. Furthermore, the OR-join $c5$ could eventually fire. Finally, the first OR-join $c1$ would have to wait for that token before firing. To put it short, if $c1$ could fire, it would have to wait. One can show that this also holds the other way around: if it

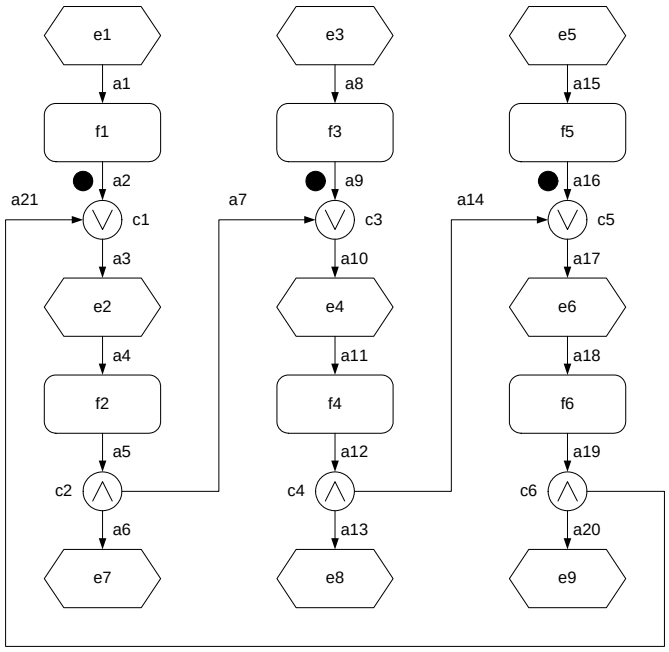


Figure 3.7: EPCs with three OR-joins on the loop

could not fire, it would not have to wait. Furthermore, this observation is also true for the two other OR-joins. In the subsequent section, we will discuss whether this problem can be resolved.

Refinement is another issue related to OR-joins. Figure 3.8 shows two versions of an EPC process model. In Figure 3.8(a) there is a token on $a7$. The subsequent OR-join $c2$ must wait for this token and synchronize it with the second token on $a5$ before firing. In Figure 3.8(b) the sequence $e3$-$a7$-$f3$ is refined with a block of two branches between an OR-split $c3a$ and an OR-join $c3b$. The OR-join $c2$ is enabled and should wait for the token on $a7f$. The question here is whether such a refinement might change the behavior of the OR-join $c1$. Figure 3.8 is just one simple example. The answer to this question may be less obvious if the refinement is introduced in a loop that already contains an OR-join. Figure 3.9 shows a respective case of an OR-join $c1$ on a loop that is refined with an OR-Block $c3a$-$c3b$. One would expect that the EPC of Figure 3.8(a) exhibits the same behavior as the one in (b). In the following section, we will discuss these questions from the perspective of different formalization approaches.

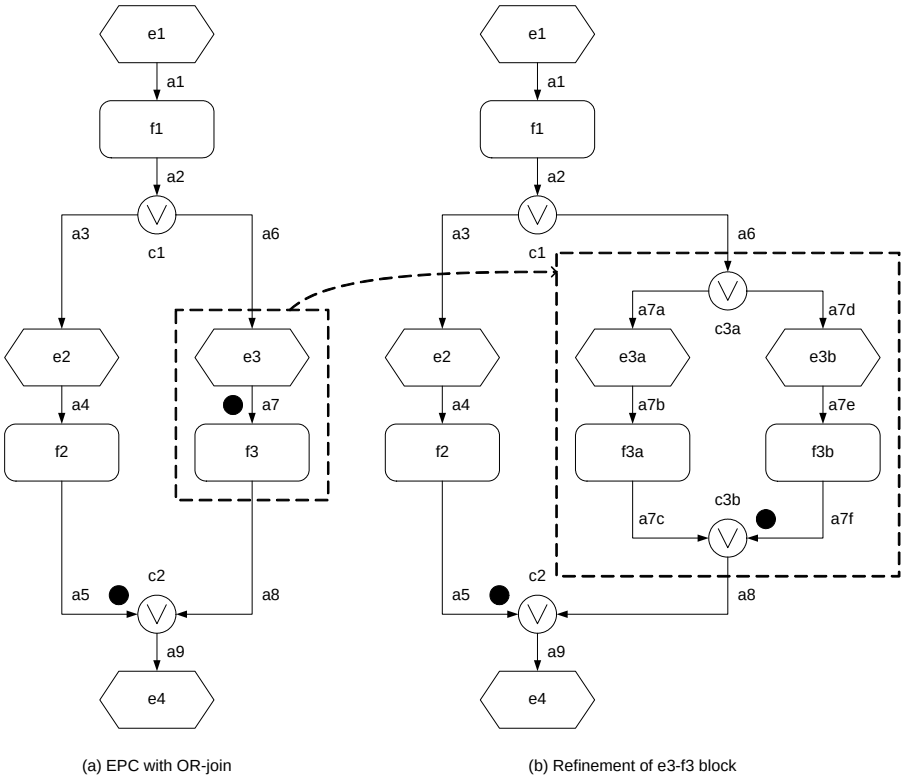


Figure 3.8: EPC refined with an OR-Block

3.4.3 Approaches to EPC Semantics Formalization

The transformation to Petri nets plays an important role in early formalizations of EPC semantics. In *Chen and Scheer* [CS94], the authors define a mapping to colored Petri nets and address the non-local synchronization behavior of OR-joins. This formalization builds on the assumption that an OR-split always matches a corresponding OR-join. The colored token that is propagated from the OR-split to the corresponding OR-join signals which combination of branches is enabled. Furthermore, the authors describe the state space of some example EPCs by giving reachability graphs. However, this first Petri net semantics for EPCs has mainly two weaknesses. Firstly, a formal algorithm to calculate the state space is missing. Secondly, the approach is restricted to EPCs with matching OR-split and OR-join pairs. Therefore, this approach does not provide semantics for the EPCs shown in figures 3.6 and 3.7. Even though the approach is not formalized in all its

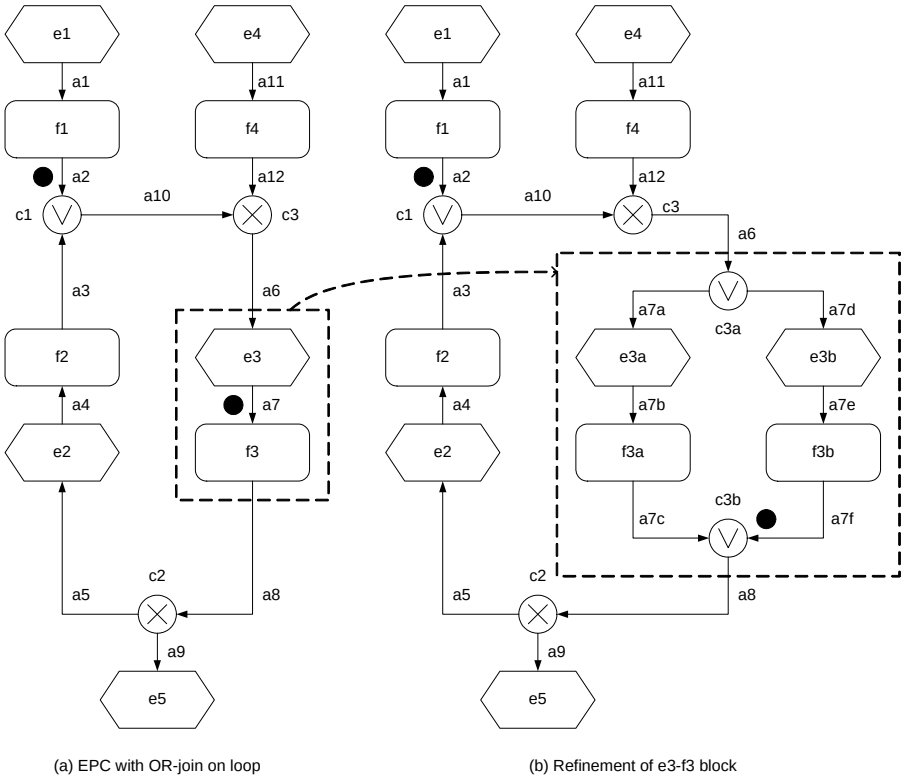


Figure 3.9: Cyclic EPC refined with an OR-Block

details, it should be able to handle the refined EPC of Figure 3.8(b) and the inner OR-join $c3b$ in Figure 3.8(b).

The transformation approach by *Langner, Schneider, and Wehler* [LSW97a, LSW97b, LSW98] maps EPCs to Boolean nets in order to define formal semantics. Boolean nets are a variant of colored Petri nets whose token colors are 0 (negative token) and 1 (positive token). Connectors propagate both negative and positive tokens according to their logical type. This mechanism is able to capture the non-local synchronization semantics of the OR-join similar to dead-path elimination in workflow systems (see [LA94, LR00]). The XOR-join only fires if there is one positive token on incoming branches and a negative token on all other incoming branches. Otherwise, it blocks. A drawback of this semantics definition is that the EPC syntax has to be restricted: arbitrary structures are not allowed. If there is a loop it must have an XOR-join as entry point and an XOR-split as exit point. This pair of connectors in a cyclic structure is mapped to one place in the resulting

Boolean net. As a consequence, this approach does not provide semantics for the EPCs in Figures 3.6 and 3.7. Still, it can cope with any pair of matching OR-split and OR-join. Accordingly, the Boolean nets define the expected semantics of the refined EPC of Figure 3.8(b) and of the inner OR-Block introduced as a refinement in Figure 3.8(b).

Van der Aalst [Aal99] presents a mapping approach to derive Petri nets from EPCs, but he does not give a mapping rule for OR-connectors because of the semantic problems illustrated in Section 3.4.2. The mapping provides clear semantics for XOR and AND-connectors as well as for the OR-split, but since the OR-join is not formalized, the approach does not provide semantics for the EPCs of Figures 3.6 to 3.9. *Dehnert* presents an extension of this approach by mapping the OR-join to a Petri net block [Deh02]. Since the resulting Petri net block may or may not necessarily synchronize multiple tokens at runtime (i.e., a non-deterministic choice), its state space is larger than the actual state space with synchronization. Based on the so-called relaxed soundness criterion, it is possible to cut away undesirable paths and, thus, check whether a join should be synchronized (cf. [DA04]).

In [Rit99, Rit00] *Rittgen* discusses the OR-join. He proposes to distinguish between three types of OR-joins on the syntactic level: every-time, first-come, and wait-for-all. The every-time OR-join basically reflects XOR-join behavior; the first-come OR-join passes the first incoming token and blocks afterwards; and the wait-for-all OR-join depends on a matching split similar to the approach of *Chen and Scheer*. This proposal could provide semantics for the example EPCs of Figures 3.6 to 3.9 in the following way. If we assume an every-time semantics, all OR-joins of the example EPCs could fire. While the loops would not block in this case, there would be no synchronization at all which contradicts the intended OR-join semantics. If the OR-joins behave according to the first-come semantics, all OR-joins could fire. Yet, there would also be no synchronization and the loops could be run only once. If the OR-joins had wait-for-all semantics, we would have the same problems as before with the loops. Altogether, the proposal by *Rittgen* does not provide a general solution to the formalization problem.

Building on prior work of *Rump* [ZR96, Rum99], *Nüttgens and Rump* [NR02] define a transition relation for EPCs that also addresses the non-local semantics of the OR-join, yet there is a problem: the transition relation for the OR-join refers to itself under

negation. *Van der Aalst, Desel, and Kindler* [ADK02] show, that a fixed point for this transition relation does not always exist. They present an example to prove the opposite: an EPC with two OR-joins on a circle, which wait for each other as depicted in Figure 3.6(b). This vicious circle is the starting point for the work of *Kindler* towards a sound mathematical framework for the definition of non-local semantics for EPCs. In a series of papers [Kin03, Kin04, Kin06], *Kindler* elaborates on this problem in detail. The technical problem is that for the OR-join the transition relation R depends upon itself in negation. Instead of defining one transition relation, he considers a pair of transition relations (P, Q) on the state space Σ of an EPC and a monotonously decreasing function $R : 2^{\Sigma \times N \times \Sigma} \rightarrow 2^{\Sigma \times N \times \Sigma}$. Then, a function $\varphi((P, Q)) = (R(Q), R(P))$ has a least fixed point and a greatest fixed point. P is called pessimistic transition relation and Q optimistic transition relation. An EPC is called *clean*, if $P = Q$. For most EPCs, this is the case. Some EPCs, such as the vicious circle EPC, are *unclean* since the pessimistic and the optimistic semantics do not coincide. Moreover, *Cuntz* provides an example of a clean EPC, which becomes unclean by refining it with another clean EPC [Cun04, p.45]. *Kindler* even shows that there are acyclic EPCs that are unclean (see [Kin06, p.38]). Furthermore, *Cuntz and Kindler* present optimizations for an efficient calculation of the state space of an EPC, and a respective prototype implementation called EPC Tools [CK04, CK05]. EPC Tools also offers a precise answer to the questions regarding the behavior of the example EPCs in Figures 3.6 to 3.9.

- Figure 3.6(a): For the EPC with one OR-join on a loop, there is a fixed point and the connector is allowed to fire.
- Figure 3.6(b): The EPC with two OR-joins on a loop is unclean. Therefore, it is not clear whether the optimistic or the pessimistic semantics should be considered.
- Figure 3.7: The EPC with three OR-joins is also unclean, i.e., the pessimistic deviates from the optimistic semantics.
- Figure 3.8(a): The OR-join $c2$ must wait for the second token on $a7$.
- Figure 3.8(b): The OR-join $c2$ must wait for the second token on $a7f$.
- Figure 3.9(a): The OR-join $c1$ must wait for the second token on $a7$.
- Figure 3.9(b): The OR-join $c1$ is allowed to fire, the second OR-join $c2$ in the OR-block must wait.

Even though the approach by *Kindler* provides semantics for a large subclass of EPCs, i.e. clean EPCs, there are some cases like the EPCs of Figure 3.6(b) and 3.7 that do not have semantics. The theorem by *Kindler* proves that it is not possible to calculate these EPCs semantics as long as the transition relation is defined with a self-reference under negation. Furthermore, such a semantics definition may imply some unexpected results, e.g. the EPC of Figure 3.9(a) behaves differently than its refinement in Figure 3.9(b).

While it is argued that unclean EPCs only have theoretical relevance, there actually are unclean EPCs in practice. Figure 3.10 shows the Test Equipment Management EPC from the Quality Management branch of the SAP Reference Model (cf. [KT98]). The marking of this EPC in the figure can be produced by firing the OR-split on the right-hand side of the EPC. Both XOR-joins are on a loop resulting in an unclean marking. This illustrates the need in theory *and* practice to also provide semantics for EPCs that are unclean, according to *Kindler's* definition [Kin06].

Van Hee, Oanea, and Sidorova discuss a formalization of extended EPCs as they are implemented in the simulation tool of the ARIS Toolset (see [IDS03a]) based on a transition system [HOS05]. This transition system is mapped to colored Petri nets in order to do verification with CPN Tools (see [JKW07]). The considered EPC extension includes data attributes, time, and probabilities which are used for the simulation in ARIS. The essential idea of this formalization and the ARIS implementation is that process folders can have timers, and that these timers are used at an OR-join for synchronization purposes.[2] If a folder arrives at an OR-join it has to wait until its timer expires. Since the timers are only reduced if there are no folders to propagate, the OR-join can synchronize multiple incoming folders. A problem of this approach is that once the timer of a folder is expired, there is no way to synchronize it once it has passed the OR-join. If there are several OR-joins used in sequence, only the first one will be synchronized. Therefore, this formalization – though being elaborate – provides only a partial solution to the formalization of the OR-join.

Van der Aalst and Ter Hofstede defined a workflow language called YAWL [AH05] which also offers an OR-join with non-local semantics. As *Mendling, Moser, and Neu-*

[2]Note that this general approach can be parameterized in ARIS with respect to sychronization and waiting semantics (see [HOS05, p.194]).

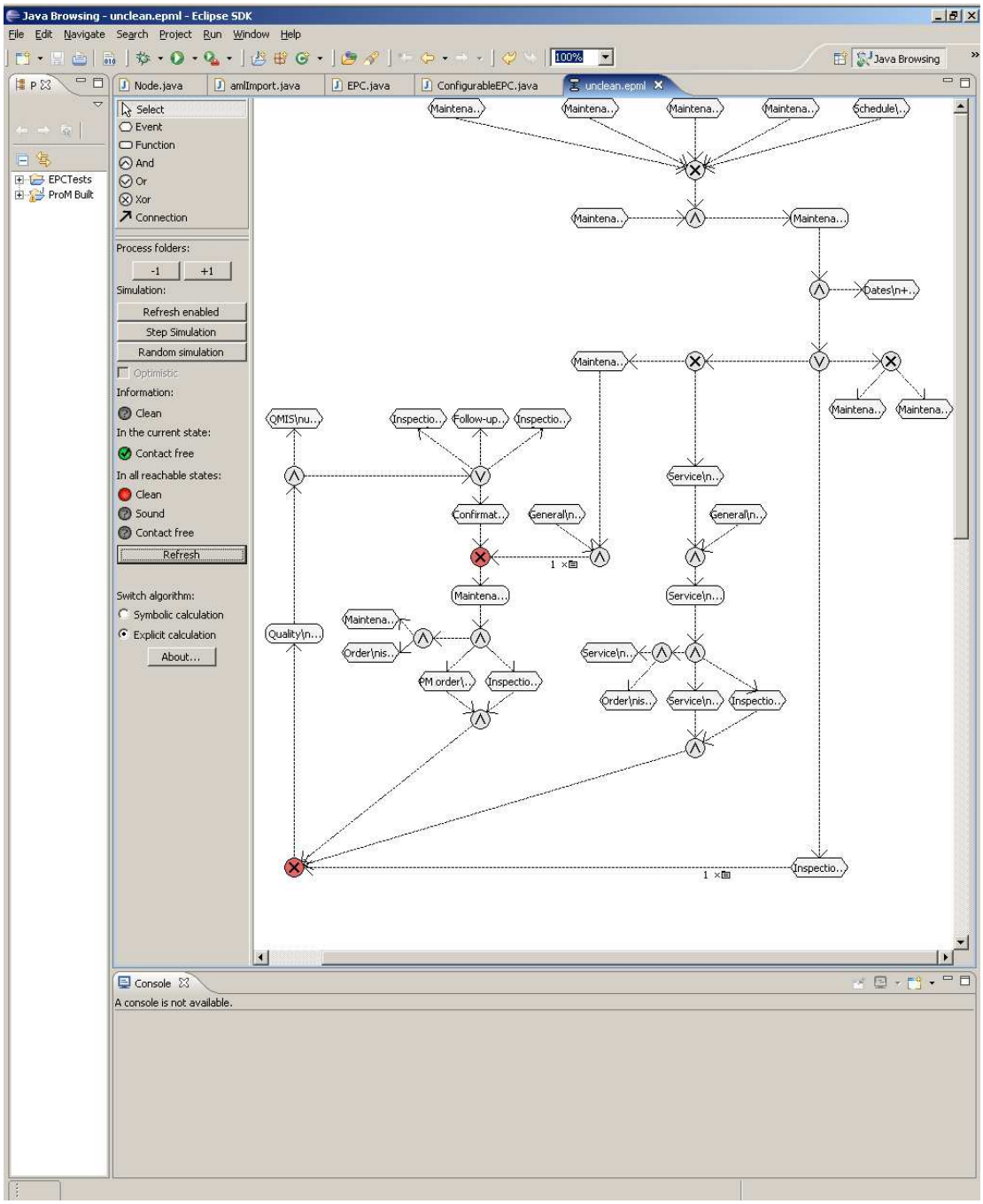


Figure 3.10: Test Equipment Management EPC from the Quality Management branch of the SAP Reference Model

mann propose a transformation semantics for EPCs based on YAWL [MMN06a], we will discuss how the OR-join behavior is formalized in YAWL. In [AH05], the authors propose a definition of the transition relation $R(P)$ with a reference to a second transition relation P that ignores all OR-joins. A similar semantics that is calculated on history-logs of the process is proposed by *Van Hee, Oanea, Serebrenik, Sidorova, and Voorhoeve* in [HOS$^+$06]. The consequence of this definition can be illustrated using the example EPCs.

- Figure 3.6(a): The single OR-join on the loop can fire.
- Figure 3.6(b): The two OR-joins on the loop can fire.
- Figure 3.7: The three OR-joins on the loop can fire.
- Figure 3.8(a): The OR-join $c2$ must wait for the second token between e3 and f3.
- Figure 3.8(b): Both OR-joins can fire.
- Figure 3.9(a): The OR-join $c1$ must wait for the second token between e3 and f3.
- Figure 3.9(b): Both OR-joins can fire.

Kindler criticizes that each choice for defining P "appears to be arbitrary or ad hoc in some way" [Kin06] and uses the pair (P, Q) instead. The example EPCs illustrate that the original YAWL semantics provide for a limited degree of synchronization. Consider, for example, the vicious circle EPC with three OR-joins: all are allowed to fire, but if one does, the subsequent OR-join has to wait. Furthermore, the refined EPCs exhibit different behavior from their unrefined counterparts since OR-joins are ignored, i.e. they are considered unable to fire.

Wynn, Edmond, Van der Aalst, and Ter Hofstede illustrate that the OR-join semantics in YAWL exhibit some non-intuitive behavior when OR-joins depend upon each other [WEAH05]. Therefore, they present a novel approach based on a mapping to Reset nets. Whether or not an OR-join can fire (i.e. $R(P)$), is determined depending on (a) a corresponding Reset net (i.e. P) that treats all OR-joins as XOR-joins[3], and (b) a predicate

[3] In fact, [WEAH05] proposes two alternative treatments for the "other OR-joins" when evaluating an OR-join: treat them either as XOR-joins (optimistic) or as AND-joins (pessimistic). However, the authors select the optimistic variant because the XOR-join treatment of other OR-joins more closely match the informal semantics of the OR-join.

called *superM* that prevents firing if an OR-join is on a directed path from another enabled OR-join. In particular, the Reset net is evaluated using backward search techniques that grant coverability to be decidable (see [LL00, FS01]). A respective verification approach for YAWL nets is presented in [WAHE06]. Using these semantics, the example EPCs behave as follows:

- Figure 3.6(a): The single OR-join on the loop can fire since *superM* evaluates to false, and hence no more tokens can arrive at c_1.
- Figure 3.6(b): The two OR-joins are not enabled since *superM* evaluates to true, because if the respectively other OR-join is replaced by an XOR-join, an additional token may arrive.
- Figure 3.7: The three OR-joins are not enabled, because if one OR-join assumes the other two to be XOR-joins, then this OR-join has to wait.
- Figure 3.8(a): The OR-join $c2$ must wait for the second token on *a7*.
- Figure 3.8(b): The OR-join $c2$ must wait for the second token on *a7f*.
- Figure 3.9(a): The OR-join *c1* must wait for the token on *a7*.
- Figure 3.9(b): The OR-join *c1* must wait because if $c3b$ is assumed to be an XOR-join a token may arrive via $a3$. The OR-join *c3b* must also wait, because if $c1$ is an XOR-join, another token may move to $a7c$. Therefore, there is a deadlock.

The novel approach based on Reset nets provides interesting semantics, but in some cases also leads to deadlocks.

Table 3.3: Overview of EPC semantics and their limitations

OR-join semantics	Limitations
[CS94]	OR-join must match OR-split
[LSW98]	Joins as loop entry undefined
[Rit99] every-time	missing synchronization
[Rit99] first-come	OR-join can block
[Rit99] wait-for-all	OR-join as loop entry undefined
[Kin06]	EPC can be unclean
[HOS05]	folders with expired timers do not synchronize
[AH05]	limited synchronization
[WAHE06]	OR-join may block

Table 3.3 summarizes existing work on the formalization of the OR-join. Several early approaches define syntactical restrictions, such as OR-splits, to match corresponding OR-joins or models to be acyclic (see [CS94, LSW98, Rit99]). Newer approaches impose little or even no restrictions (see [Kin06, AH05, WAHE06]), but exhibit unexpected behavior for OR-block refinements on loops with further OR-joins on it. The solution based on Reset nets seems to be most promising from the intuition of its behavior. Yet, it requires extensive calculation effort since it depends upon backward search to decide coverability (Note that reachability is undecidable for reset nets illustrating the computational complexity of the OR-join in the presence of advanced routing constructs). In the following subsection, we propose a novel approach that overcomes some of the refinement problems of the Reset nets semantics and that provides a more intuitive solution since all OR-join decisions can be made with local knowledge.

3.4.4 A Novel Approach towards EPC Semantics

In this subsection, we introduce a novel concept for EPC semantics.[4] The formalization of this concept follows in the subsequent section. The principal idea of these semantics borrows some concepts from *Langner, Schneider, and Wehler* [LSW98] and adapts the idea of Boolean nets with true and false tokens in an appropriate manner. Furthermore, we utilize similar notations as *Kindler* [Kin06], modifying them where needed. The transition relations that we will formalize afterwards depend on the state and the context of an EPC. The *state* of an EPC is basically an assignment of positive and negative tokens to the arcs. Positive tokens signal which functions have to be carried out in the process, negative tokens indicate which functions are to be ignored for the moment. The transition rules of AND-connector and OR-connectors are adopted from the Boolean nets formalization which facilitates synchronization of OR-joins in structured blocks. In order to allow for a more flexible utilization of XOR-connectors in a cyclic structure, we modify and extend the approach of Boolean nets in three ways:

[4] An earlier version of these semantics is described in [MA06]. Essentially, this version is different in two ways: (1) Dead context is propagated already if only one input is dead. Without that, XOR-loops could not be marked dead. (2) We introduce a concept to clean up negative tokens that could not be forwarded to an OR-join (see *negative upper corona* in phase 4 for positive token propagation).

1. XOR-splits produce one positive token on one of their their output arcs, but no negative tokens. XOR-joins fire each time there is a positive token on an incoming arc. This mechanism provides the expected behavior in both structured XOR-loops and structured XOR-blocks where an XOR-split matches an XOR-join.
2. In order to signal OR-joins that it is not possible to have a positive token on an incoming branch, we define the *context* of an EPC. The context assigns a status of *wait* or *dead* to each arc of an EPC. A wait context indicates that it is still possible that a positive token might arrive; a dead context status means that either a negative token will arrive next, or no positive token can arrive anymore. For example, XOR-splits produce a dead context on those output branches that are not taken, and a wait context on the output branch that receives a positive token. A dead context at an input arc is then used by an OR-join to determine whether it has to synchronize with further positive tokens or not. Since dead and wait context might be conflicting and, thus, have to alternate, both context and state is propagated in separate phases to guarantee termination.
3. The propagation of context status and state tokens is arranged in a four phase cycle: (a) dead context, (b) wait context, (c) negative token, and (d) positive token propagation.

   (a) In this phase, all *dead context* information is propagated in the EPC until no new dead context can be derived.

   (b) Then, all *wait context* information is propagated until no new wait context can be derived. It is necessary to have two phases (i.e., first the dead context propagation and then the wait context propagation) in order to avoid infinite cycles of context changes (details below).

   (c) After that, all *negative tokens* are propagated until no negative token can be propagated anymore. This phase cannot run into an endless loop (details below).

   (d) Finally, one of the enabled nodes is selected and propagates *positive tokens* leading to a new iteration of the four phase cycle.

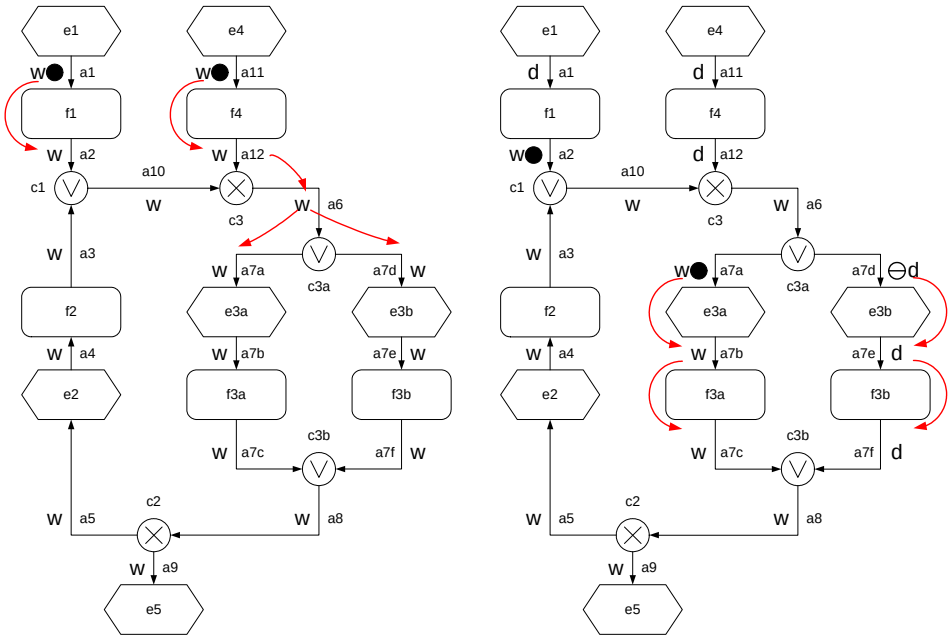
In the following part, we first give an example to illustrate the behavior of the EPC semantics before defining state, context, and each transition phase in detail.

Revisiting the cyclic EPC refined with an OR-block

Figure 3.11 revisits the example of the cyclic EPC refined with an OR-block that we introduced as Figure 3.9 in Section 3.4.2.

In Figure 3.11(a), an initial marking with two positive tokens on $a1$ and $a11$ is given. These positive tokens induce a wait context on all arcs, which implies that all of them might potentially receive a positive token at some point in time. The context status is indicated by a letter next to the arc: w for wait and d for dead. Subsequently, the positive tokens can be propagated to the arcs $a2$ and $a12$, respectively and the context of $a1$ and $a11$ changes to dead. In this situation, the OR-join $c1$ is not allowed to fire due to the wait context on arc $a3$, but has to synchronize with positive and negative tokens that might arrive there. The XOR-join is allowed to fire without considering the second arc $a10$. In (b) the OR-split $c3a$ has fired (following the execution of c3) and produces a positive token on $a7a$ and a negative token on $a7d$. Accordingly, the context of $a7d$ is changed to dead. This dead context is propagated down to arc $a7f$. The rest of the context remains unchanged. The state shown in (b) is followed by (c) where the positive and the negative tokens are synchronized at the connector $c3b$ and one positive token is produced on the output arc $a8$. Please note that the OR-join $c3b$ does not synchronize with the other OR-join $c1$ that is also on the loop. In the *Kindler* and the Reset nets semantics, $c3b$ would have to wait for the token from $a2$. Here, the wait context propagation is blocked by the negative token. In (d), the XOR-split $c2$ produces a positive token on $a9$ and a dead context on $a5$. This dead context is propagated via $a3$ to the rest of the loop in the dead context propagation phase. In the wait context propagation phase, the dead context of the loop is reset to wait, which is propagated from $c1$. As a consequence, the OR-join $c1$ is not enabled.

This example allows us to make two observations. Firstly, the context propagation blocks OR-joins that are entry points to a loop in a wait position since the self-reference is not resolved. Secondly, the XOR-split produces a dead context, but not a negative



(a) Initial Marking

(b) Entering the OR-block

(c) Synchronizing

(d) Exiting the loop

Figure 3.11: Example of EPC marking propagation

token. The disadvantage of producing negative tokens would be that the EPC is flooded with negative tokens if an XOR-split was used as an exit of a loop. These tokens would give downstream joins the wrong information about the state of the loop, since it would still be live. An OR-join could then synchronize with a negative token while a positive token is still in the loop. In contrast to that, the XOR-split as a loop exit produces a dead context. Since there is a positive token in the loop, it overwrites the dead context at the exit in the wait context propagation phase. Downstream OR-joins then have the correct information that there are still tokens to wait for.

Definition of State, Context, and Marking

We define both state and context as an assignment to the arcs. The term *marking* refers to state and context together. The EPC transition relations defines which state and/or context changes are allowed for a given marking in a given phase.

Definition 3.12 (State and Context)**.** Let $EPC = (E, F, C, l, A)$ be a standard EPC. Then, a mapping $\sigma : A \to \{-1, 0, +1\}$ is called a state of an EPC. The positive token captures the state as it is observed from outside the process. It is represented by a black filled circle. The negative token depicted by a white open circle with a minus on it has a similar semantics as the negative token in the Boolean nets formalization. Arcs with no state tokens on them do not depict a circle. Furthermore, a mapping $\kappa : A \to \{wait, dead\}$ is called a context of an EPC. A wait context is represented by a w and a dead context by a d next to the arc.

In contrast to Petri nets, we distinguish the terms *marking* and *state*: the term marking refers to state σ and context κ collectively.

Definition 3.13 (Marking of an EPC)**.** Let $EPC = (E, F, C, l, A)$ be a standard EPC. Then, a mapping $m : A \to (\{-1, 0, +1\} \times \{wait, dead\})$ is called a marking. The set of all markings M_{EPC} of an EPC is called marking space with $M_{EPC} = A \times \{-1, 0, +1\} \times \{wait, dead\}$. The projection of a given marking m to a subset of arcs $S \subseteq A$ is referred to as m_S. The marking m_a of an arc a can be written as $m_a = (\kappa(a) + \sigma(a)) \cdot a$, e.g. $(w + 1)a$ for an arc with a wait context and a positive token. If we refer to the κ- or the σ-part of m, we write κ_m and σ_m, respectively, i.e. $m(a) = (\sigma_m(a), \kappa_m(a))$.

Initial and Final Marking

The initial marking is the starting point for applying an iteration of the four-phase cycle. In [Rum99], the initial marking of an EPC is specified as an assignment of tokens to one, some, or all start events. While such a definition contains enough information for verification purposes, for example by the bundling of start and end events with OR-connectors as proposed in [MMN06a], it does not provide executable semantics according to the original definition of EPCs. As pointed out in [Rit99], it is not possible to equate the triggering of a single start event with the instantiation of a new process. This is because EPC start events do not only capture the creation of a process instance, but also external events that influence the execution of a running EPC (cf. [CS94]). This observation suggests an interactive validation approach as presented by [DAV05], where the user makes explicit assumptions about potential combinations of start events. In our approach, we assume that in the initial marking, all start arcs $a_s \in A_s$ have either a positive or a negative token with the matching context[5]. A respective formalization of initial and final marking is given later in Definitions 3.14 and 3.15. In the following sections, we describe the transition relations of each node $n \in E \cup F \cup C$ in the phases of dead context, wait context, negative and positive token propagation.

Phase 1: Dead Context Propagation

The transition relation for dead context propagation defines rules for deriving a dead context if one input arc of a node has a dead context status. Note that this rule might result in arcs having a dead context that could still receive a positive token. Those arcs are reset to a wait context in the subsequent phase of wait context propagation (Phase 2).

Figure 3.12 gives an illustration of the transition relation. *Please note that the figure does not depict the fact that the the rules for dead context propagation can only be applied if the respective output arc does not hold a positive or a negative token.* Concrete tokens override context information, for isntance, an arc with a positive token will always

[5]The context of non-start arcs is derived when the four propagation phases are entered the first time. We choose to initialize all non-start arcs with a wait context (cf. Figure 3.11). Note that this context might be changed in the dead context propagation phase before any token is moved.

have a wait context. Rules (a) and (b) indicate that if an input arc of a function or an event is dead, then also the output arc has to have a dead context status. Rule (c) represents that each split-connector propagates a dead context to its output arcs. These transition relations formalize the observation that if an input arc cannot receive a token anymore, this also holds true for its output arcs (unless they already hold positive or negative tokens). The join-connectors require only one dead context status at their input arcs for reproducing it at their output arc, see (d). It is important to note that a dead context is propagated until there is an end arc or an arc that carries a token.

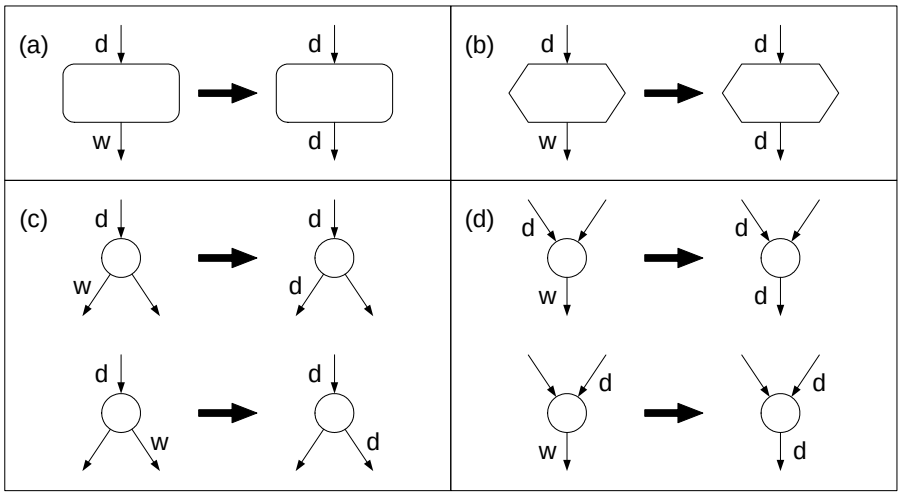


Figure 3.12: Transition relation for dead context propagation

Phase 2: Wait Context Propagation

The transition relation for wait context propagation defines rules for deriving a wait context if one or more input arcs of a node have a wait context status. Figure 3.13 gives an illustration of the transition relation. *All transitions can only be applied if the respective output arc does not hold a positive or a negative token.* Concrete tokens override context information, i.e. an arc with a positive token will always have a wait context. Rules (a) and (b) show that if an input arc of a function or an event has a wait context, then the output arc also has to have a wait context status. Rule (c) represents that each split-connector propagates a wait context to its output arcs. The AND-join requires all inputs to have a

wait context status in order to reproduce it at its output arc, see (d). XOR- and OR-joins propagate a wait context if one of their input arcs has a wait context, see (e) and (f). Similar to the dead context propagation, the wait context is propagated until an end node is received or until an arc holds a token. Furthermore, the wait context is propagated by an AND-join where all of the inputs have a wait context.

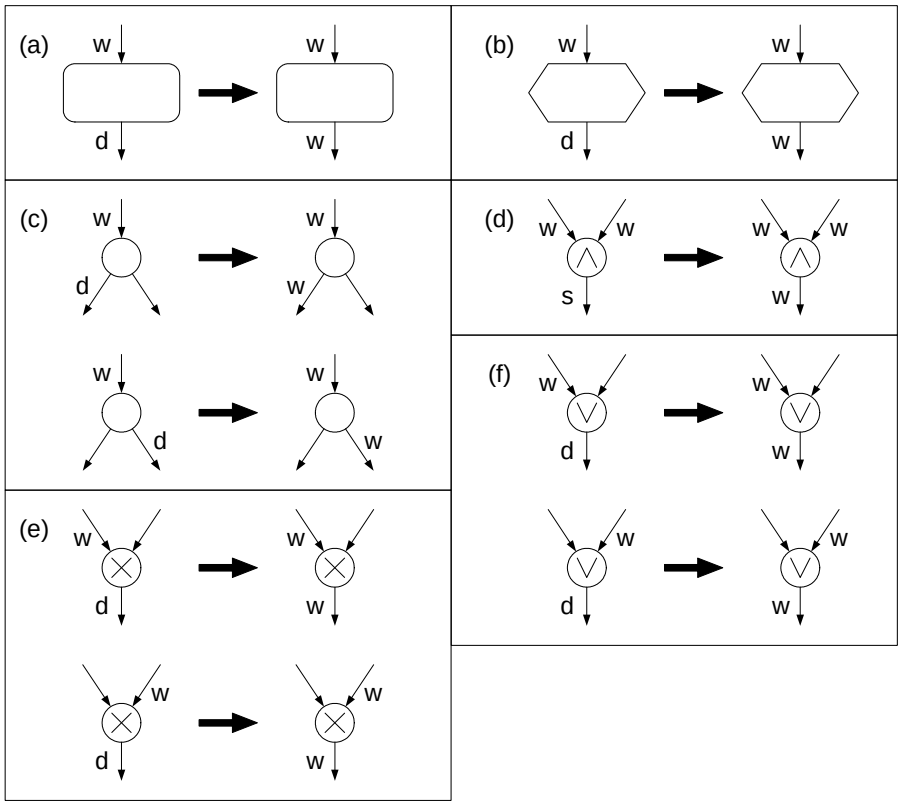


Figure 3.13: Transition relation for wait context propagation

Observations on Context Propagation

The transition relations of context propagation permit the following observations:

- *Context changes terminate:* It is intuitive that context propagation cannot run in an infinite loop. It is easy to verify that the first two phases stop. The propagation of

dead context stops because the number of arcs is finite, i.e., the number of arcs is an upper bound for the number of times the rules in Figure 3.12 can be applied. A similar argument applies to the propagation of the wait context. As a consequence, the context change phase will always terminate and enable the consideration of new state changes in the subsequent phase.

- *State tokens block context propagation:* The transition relations for context propagation require that the output arcs to be changed do not hold any state token, i.e., arcs with a positive token always have a wait context, and arcs with a negative token always have a dead context.
- *Context propagating elements:* Functions, events, and split nodes reproduce the context that they receive at their input arcs.
- *OR- and XOR-joins:* Both these connectors reproduce a dead and also a wait context if at least one of the input arcs has the respective context.
- *AND-joins:* AND-joins produce wait context status only if all inputs are wait. Otherwise, the output context remains in a dead context.

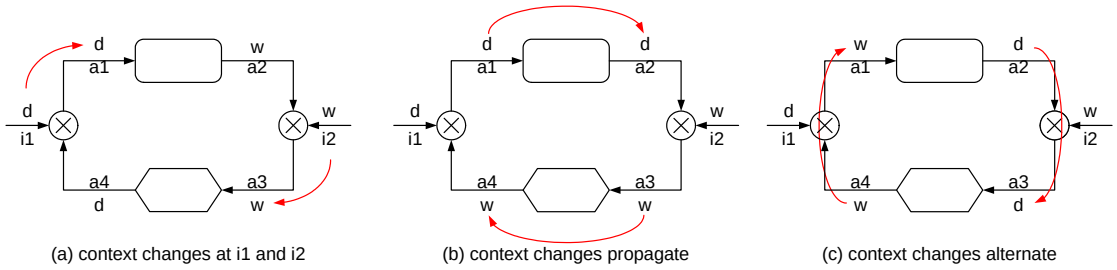


Figure 3.14: Situation of unstable context changes without two phases

Figure 3.14 illustrates the need to perform context propagation in two separate phases as opposed to together in one phase. If there are context changes (a) at $i1$ and $i2$, the current context enables the firing of the transition rules for both connectors producing a $dead$ context status in $a1$ and a $wait$ context status in $a3$. This leads to a new context in (b) with an additional $dead$ context status in $a2$ and a new $wait$ context status in $a4$. Since both arcs from outside the loop to the connectors are marked in such a way that incoming context changes on the other arc is simply propagated, there is a new context in (c) with a $wait$ status in $a1$ and a $dead$ context status in $a3$. Note that this new context can be

propagated, and this way the initial situation is reproduced. This can be repeated again and again. Without a sequence of two phases, the transitions could continue infinitely and the result would be undefined.

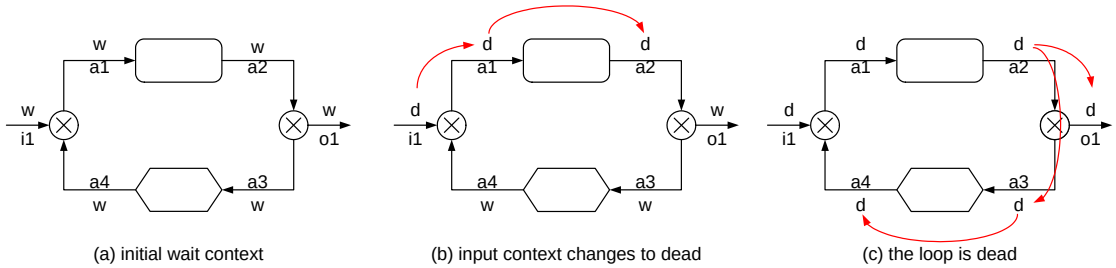


Figure 3.15: Propagating dead context in a loop

The precedence of the two phases can also be motivated using an example EPC containing a loop. The propagation of dead context with only one dead input is needed to accurately mark loops as dead. Figure 3.15 shows the picture of a simple loop with one XOR-join as entrance and one XOR-split as exit. Initially, the loop might be in a wait context (a). If the path to the loop becomes dead, this context is propagated into the loop (b) and to its output (c). If not all join-connectors would propagate dead context with one dead input, the loop could never become dead. But since this often results in too many dead arcs, the wait context propagation must be performed afterwards. It guarantees that arcs that can still be receive a positive token get a wait context.

Phase 3: Negative Token Propagation

Negative tokens can result from branches that are not executed after OR-joins or start events. The transition relation for negative token propagation includes four firing rules that consume and produce negative tokens. Furthermore, the output arcs are set to a dead context. Figure 3.16 gives an illustration of the transition relation. *All transitions can only be applied if all input arcs hold negative tokens and if there is no positive token on the output arc.* In the following section, we will show that this phase terminates.

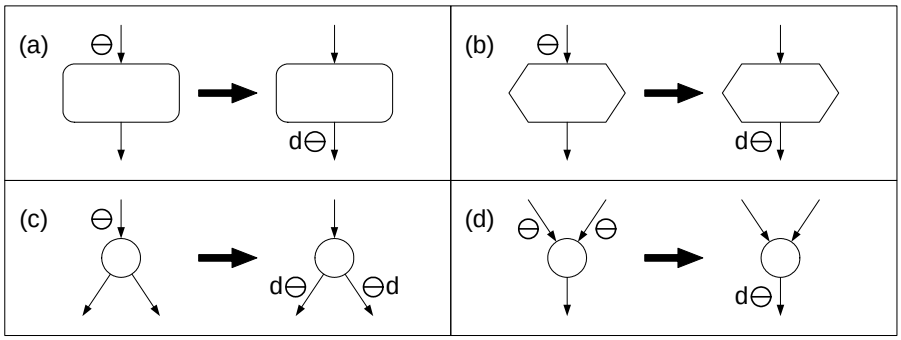


Figure 3.16: Transition Relation for Negative Token Propagation

Phase 4: Positive Token Propagation

The transition relation for positive token propagation specifies firing rules that consume negative and positive tokens from the input arcs of a node to produce positive tokens on its output arcs. Figure 3.17 gives a respective illustration. Rules (a) and (b) show that functions and events consume positive tokens from the input arc and propagate them to the output arc. Furthermore, and this holds true for all rules, consuming a positive token from an arc implies setting this arc to a dead context status. Rules (c) and (d) illustrate that AND-splits consume one positive token and produce one on each output arc, while AND-joins synchronize positive tokens on all input arcs to produce one on the output arc. Rule (e) depicts the fact that XOR-splits forward positive tokens to one of their output arcs. In contrast to the Boolean net formalization, they do not produce negative tokens, but a dead context on the output arcs which do not receive the token. Correspondingly, XOR-joins (f) propagate each incoming positive token to the output arc, no matter what the context or the state of the other input arcs is. If there are negative tokens on the incoming arcs, they are consumed. The OR-split (g) produces positive tokens on those output arcs that have to be executed and negative tokens on those that are ignored. Note that the OR-join is the only construct that may introduce negative tokens (apart from start events that hold a negative token in the initial marking). Rule (h) shows that on OR-join can only fire either if it has full information about the state of its input arcs (i.e., each input has a positive or a negative token) or all arcs that do not hold a token are in a dead context. Finally, in all rules, each output arc that receives a negative token is set to a dead

context and each that gets a positive token is set to a wait context.

The last two firing rules of the OR-join in Figure 3.17(h) deserve some further comments. Beyond the removal of all positive and negative tokens on the input arcs, also those negative tokens on the *negative upper corona* of the OR-join are removed. The motivation for this concept is that loops can propagate dead context, but negative tokens get stuck at the entry join of a loop. After the loop, a dead context can make the firing condition of an OR-join become true, while negative tokens that were generated for synchronization purposes still reside before the loop. Not removing such negative tokens with the firing of an OR-join might cause non-intuitive behavior. Therefore, in addition to the positive and negative tokens on the input arcs of the OR-join, also those negative tokens with a path leading to the OR-join via arcs that all have a dead context, i.e. on the negative corona, are also removed.

Figure 3.18 gives the example of a structured EPC with an outer XOR-loop between $c1$ and $c6$ and an inner XOR-loop between $c3$ and $c4$. The inner loop is also nested in an OR-block between $c2$ and $c5$. The current marking is produced by firing the OR-split with a negative token to the left and a positive token to the right, and then propagating the positive token via $f2$. Now, the OR-join $c5$ is enabled with a dead context on one of the input arcs. Moreover, there is a negative token before the inner XOR-loop which cannot be propagated. If the OR-join would now simply fire and navigate via $e2$ back to $c2$ the EPC would be in a deadlock since the firing rules for tokens require the output arcs to be empty. Therefore, the negative token before $c3$ has to be removed when firing the OR-join $c5$. Accordingly, if an OR-join fires, it has to remove all negative tokens on its so-called negative upper corona, i.e. the arcs carrying a negative token that have a path to the OR-join on which each arc has a dead context and no token on it.

Observations on State Propagation

The transition relations of state propagation permit the observation that the EPC semantics are *safe*, i.e. it is not possible to have more than one token on an arc. Firstly, this property is enforced by the definition of state since it is a mapping from the arcs to the set of -1,0, and +1. Furthermore, the state propagation rules guarantee safeness, too,

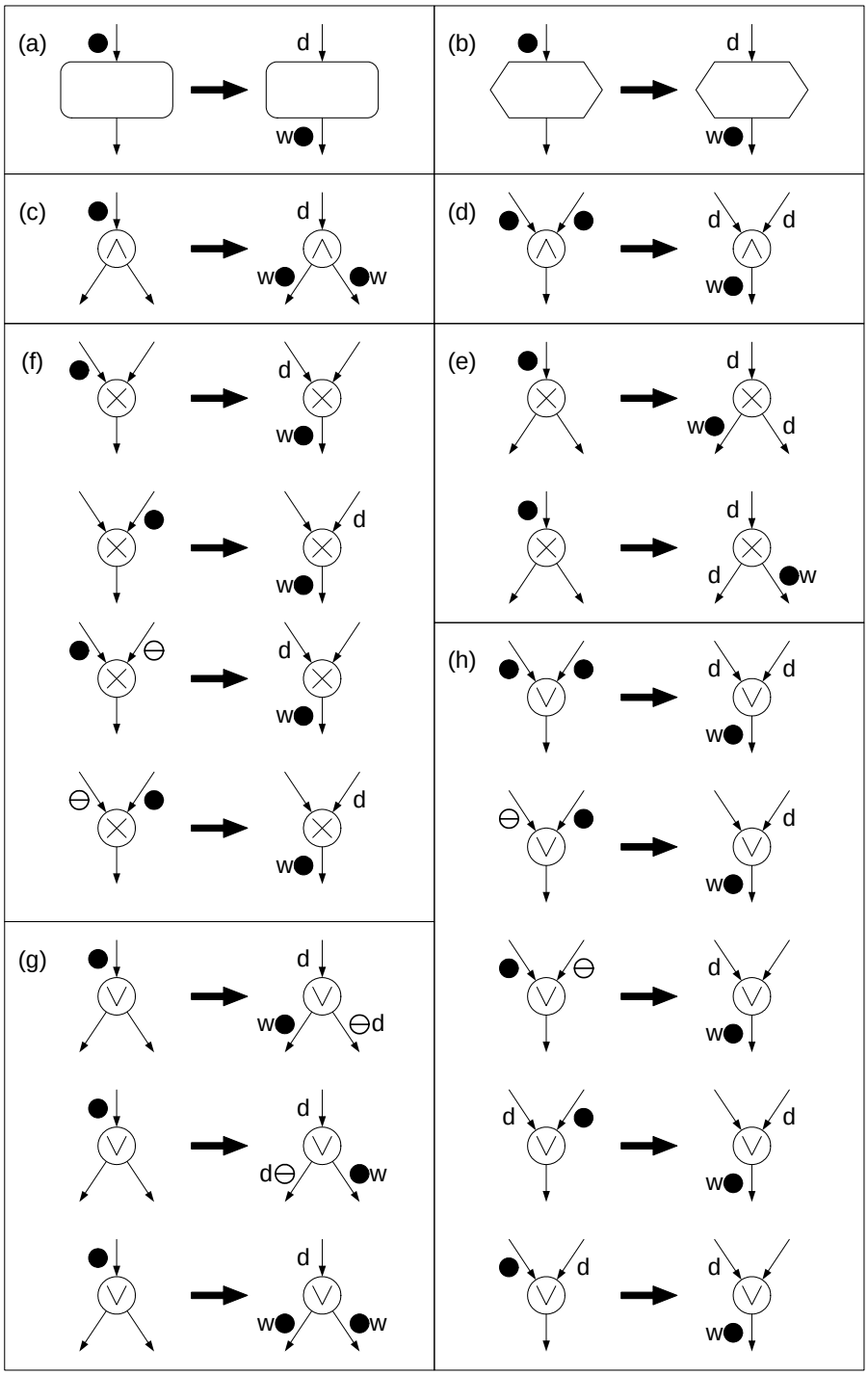


Figure 3.17: Transition Relation for Positive Token Propagation

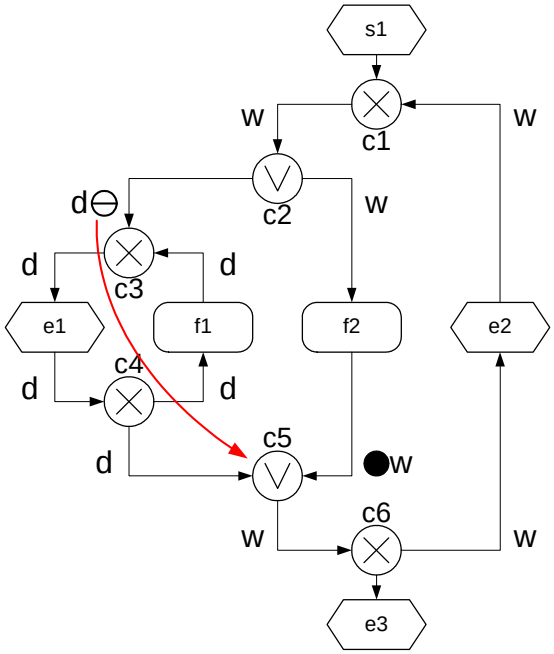


Figure 3.18: A structured EPC with a negative token on the negative upper corona of OR-join *c5*

since a node can fire only if all its outputs are empty. Due to the safeness property, we already know that the state space is finite since also the number of arcs is finite for an EPC. Another observation is that there are several state and context propagations that are not interesting to the user of the model. Therefore, the following section will make a distinction between the *transition relation* of an EPC that covers all state and context changes, and the *reachability graph* that only covers the propagation of positive tokens and hides context and negative token propagation.

This semantics definition based on state and context implies that the examples of Section 3.4 behave as follows.

- Figure 3.6(a): The single OR-join on the loop produces a wait context at $a9$. Therefore, it is blocked.
- Figure 3.6(b): The two OR-joins produce a wait context at $a23$ and $a24$. Therefore, they are both blocked.

- Figure 3.7: The three OR-joins are blocked due to a wait context at $a7$, $a14$, and $a21$.
- Figure 3.8(a): The OR-join $c2$ must wait for the second token on *a7*.
- Figure 3.8(b): The OR-join $c2$ must wait for the second token on *a7f*.
- Figure 3.9(a): The OR-join *c1* must wait for the token on *a7*.
- Figure 3.9(b): The OR-join *c1* must wait for the token on *a7*. The OR-split $c3a$ produces a negative token on $a7c$ so that $c3b$ can fire.

It can be seen that the refined EPCs exhibit the expected behavior similar to the unrefined cases, i.e. the OR-join in the structured block does not deadlock. Furthermore, if there is an OR-join as an entry point to a loop, it will deadlock if there is not a second XOR-entry that can propagate a token into this loop.

3.4.5 Transition Relation and Reachability Graph of EPCs

In this section, we formalize the concepts that were introduced in the previous section. In particular, we define the transition relations for each phase and the reachability graph of EPCs based on markings, i.e. state and context mappings σ and κ collectively. The reachability graph hides the transitions of the context propagation and negative token propagation phases. First, we provide definitions for marking, initial marking, and final marking. Then, we define the transition relations R^d, R^w, R^{-1}, and R^{+1} of an EPC for each of the four phases. Finally, we define the reachability graph RG based on the transition relations and an algorithm to calculate RG. *Please note that all definitions are applicable for relaxed syntactically correct EPCs* (see Definition 3.8 on page 47).

Definition of Initial and Final Marking

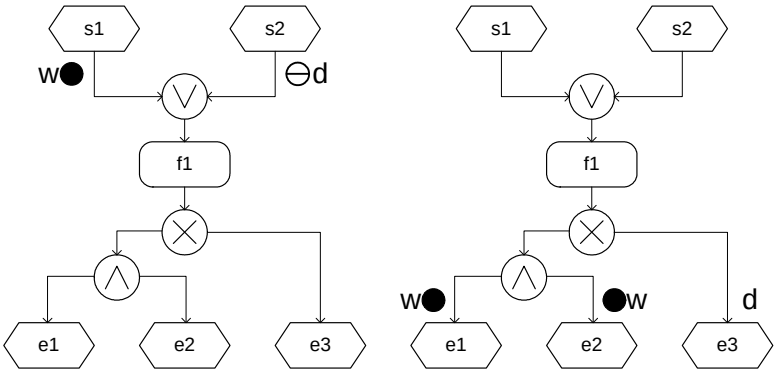
In this paragraph we define the sets of the initial and the final markings of an EPC similar to the definition in *Rump* [Rum99]. An initial marking is an assignment of positive or negative tokens to all start arcs while all other arcs have no token, and in a final marking only end arcs may hold positive tokens.

Definition 3.14 (Initial Marking of an EPC)**.** Let $EPC = (E, F, C, l, A)$ be a relaxed syntactically correct EPC and M_{EPC} its marking space. $I_{EPC} \subseteq M_{EPC}$ is defined as the set of all possible initial markings, i.e. $m \in I_{EPC}$ if and only if [6]:

- $\exists a_s \in A_s : \sigma_m(a_s) = +1$,
- $\forall a_s \in A_s$: $\sigma_m(a_s) \in \{-1, +1\}$,
- $\forall a_s \in A_s$: $\kappa_m(a_s) = wait$ if $\sigma_m(a_s) = +1$ and $\kappa_m(a_s) = dead$ if $\sigma_m(a_s) = -1$, and
- $\forall a \in A_{int} \cup A_e : \kappa_m(a) = wait$ and $\sigma_m(a) = 0$.

Definition 3.15 (Final Marking of an EPC)**.** Let $EPC = (E, F, C, l, A)$ be a relaxed syntactically correct EPC and M_{EPC} its marking space. $O_{EPC} \subseteq M_{EPC}$ is defined as the set of all possible final markings, i.e. $m \in O_{EPC}$ if and only if:

- $\exists a_e \in A_e$: $\sigma_m(a_e) = +1$ and
- $\forall a \in A_s \cup A_{int} : \sigma_m(a) \leq 0$.



(a) One particular initial marking (b) One particular final marking

Figure 3.19: Initial and final marking of an EPC

Initial and final markings are the start and end points for calculating the transition relation of an EPC. Figure 3.19(a) illustrates one particular initial marking $i \in I$ which

[6]Note that the marking is given in terms of arcs. Intuitively, one can think of start events holding positive or negative tokens. However, the corresponding arc will formally represent this token.

assigns a positive token to the left start arc and a negative token to the right start arc. The OR-join synchronizes both these tokens and may produce (after some steps) the marking that is depicted in Figure 3.19(b). There, the left branch of the XOR-split has been taken which results in positive tokens on the end arcs after the AND-split and a dead context on the right end arc.

Phase 1: Transition Relation for Dead Context Propagation

Given these definitions related to the marking of an EPC, we define the transition relations for each phase. We can summarize the different rules of Figure 3.12 in a single one: if one input arc of the respective node has a dead context, then this is propagated to the output arcs.

Definition 3.16 (Transition Relations for Dead Context Propagation)**.** Let $EPC = (E, F, C, l, A)$ be a relaxed syntactically correct EPC, $N = E \cup F \cup C$ its set of nodes, and M_{EPC} its marking space. Then $R^d \subseteq M_{EPC} \times N \times M_{EPC}$ is the transition relation for dead context propagation and $(m, n, m') \in R^d$ if and only if:

$$
\begin{aligned}
&(\exists_{a \in n_{in}} : \kappa_m(a) = dead) \wedge \\
&(\forall_{a \in A} : \sigma_m(a) = \sigma_{m'}(a)) \wedge \\
&(\exists_{X \neq \emptyset} : X = \{a \in n_{out} \mid \sigma_m(a) = 0 \ \wedge \kappa_m(a) = wait\} \wedge \\
&\qquad (\forall_{a \in X} : \kappa_{m'}(a) = dead) \wedge \\
&\qquad (\forall_{a \in A \setminus X} : \kappa_{m'}(a) = \kappa_m(a))
\end{aligned}
$$

Furthermore, we define the following notations:

- $m_1 \xrightarrow[d]{n} m_2$ if and only if $(m_1, n, m_2) \in R^d$. We say that in the dead context propagation phase marking m_1 enables node n and its firing results in m_2.
- $m \underset{d}{\longrightarrow} m'$ if and only if $\exists n : m_1 \xrightarrow[d]{n} m_2$.
- $m \xrightarrow[d]{\tau} m'$ if and only if $\exists_{n_1,\ldots,n_q,m_1,\ldots,m_{q+1}} : \tau = n_1 n_2 \ldots n_q \in N* \ \wedge$ $m_1 = m \wedge m_{q+1} = m' \wedge m_1 \xrightarrow[d]{n_1} m_2, m_2 \xrightarrow[d]{n_2} \ldots \xrightarrow[d]{n_q} m_{q+1}$.
- $m \xrightarrow[d]{*} m'$ if and only if $\exists_\tau : m \xrightarrow[d]{\tau} m'$.

- $m \overset{max}{\underset{d}{\rightarrow}} m'$ if and only if $\exists_\tau : m \overset{\tau}{\underset{d}{\rightarrow}} m' \wedge \nexists_{m'' \neq m'} : m' \underset{d}{\rightarrow} m''$.
- $max_d : M_{EPC} \rightarrow M_{EPC}$ such that $max_d(m) = m'$ if and only if $m \overset{max}{\underset{d}{\rightarrow}} m'$. The existence of a unique $max_d(m)$ is the subject of Theorem 3.1 below.

Theorem 3.1 (Dead Context Propagation terminates)**.** *For an EPC and a given marking m, there exists a unique $max_d(m)$ which is determined in a finite number of propagation steps.*

Proof. Regarding *uniqueness*, by contradiction: Consider an original marking $m_0 \in M_{EPC}$ and two markings $m_{max1}, m_{max2} \in M_{EPC}$ such that $m_0 \overset{max}{\underset{d}{\rightarrow}} m_{max1}$, $m_0 \overset{max}{\underset{d}{\rightarrow}} m_{max2}$, and $m_{max1} \neq m_{max2}$. Since both m_{max1} and m_{max2} can be produced from m_0 they share at least those arcs with dead context that were already dead in m_0. Furthermore, following from the inequality, there must be an arc a that has a wait context in one marking, but not in the other. Let us assume that this marking is m_{max1}. But if $\exists \tau : m_0 \overset{\tau}{\underset{d}{\rightarrow}} m_{max2}$ such that $\kappa_{m_{max2}}(a) = dead$, then there must also $\exists \tau' : m_{max1} \overset{\tau'}{\underset{d}{\rightarrow}} m'$ such that $\kappa_{m'}(a) = dead$ because m_{max2} is produced applying the propagation rules without ever changing a dead context to a wait context. Accordingly, there are further propagation rules that can be applied on m_{max1} and the assumption $m_0 \overset{max}{\underset{d}{\rightarrow}} m_{max1}$ is wrong. Therefore, if there are two m_{max1} and m_{max2}, they must have the same set of arcs with dead context, and therefore also the same set of arcs with wait context. Since both their states are equal to the state of m_0 they are equivalent, i.e., $max_d(m)$ is unique.
Regarding *finiteness*: Following Definition 3.11 on page 50, the number of nodes of an EPC is finite, and therefore the set of arcs is also finite. Since the number of dead context arcs is increased in each propagation step, no new propagation rule can be applied, at the latest after each arc has a dead context. Accordingly, dead context propagation terminates at the latest after $|A|$ steps. □

Phase 2: Transition Relation for Wait Context Propagation

For the wait context propagation, we also distinguish two cases based on the different transition relations of Figure 3.13. The first case covers (a) function, (b) intermediate event, (c) split, (d) and-join nodes. If the node belongs to this group and all input arcs

are in a wait context, then the wait context is propagated to those output arcs that have a dead context and no state token on them. The second case, if the node is an XOR-join or an OR-join and one of the input arcs is in a wait context, then this is propagated to the dead output arc.

Definition 3.17 (Transition Relations for Wait Context Propagation)**.** Let $EPC = (E, F, C, l, A)$ be a relaxed syntactically correct EPC, $N = E \cup F \cup C$ its set of nodes, and M_{EPC} its marking space. Then $R^w \subseteq M_{EPC} \times N \times M_{EPC}$ is the transition relation for wait context propagation and $(m, n, m') \in R^w$ if and only if:

$$
\begin{aligned}
&((n \in F \cup E_{int} \cup S \cup J_{and}) \wedge \\
&(\forall_{a \in n_{in}} : \kappa_m(a) = wait) \wedge \\
&(\forall_{a \in A} : \sigma_m(a) = \sigma_{m'}(a)) \wedge \\
&(\exists_{X \neq \emptyset} : X = \{a \in n_{out} \mid \sigma_m(a) = 0 \ \wedge \kappa_m(a) = dead\} \wedge \\
&\qquad (\forall_{a \in X} : \kappa_{m'}(a) = wait) \wedge \\
&\qquad (\forall_{a \in A \setminus X} : \kappa_{m'}(a) = \kappa_m(a))) \\
&\vee \\
&((n \in J_{xor} \cup J_{or}) \wedge \\
&(\exists_{a \in n_{in}} : \kappa_m(a) = wait) \wedge \\
&(\forall_{a \in A} : \sigma_m(a) = \sigma_{m'}(a)) \wedge \\
&(\exists_{X \neq \emptyset} : X = \{a \in n_{out} \mid \sigma_m(a) = 0 \ \wedge \kappa_m(a) = dead\} \wedge \\
&\qquad (\forall_{a \in X} : \kappa_{m'}(a) = wait) \wedge \\
&\qquad (\forall_{a \in A \setminus X} : \kappa_{m'}(a) = \kappa_m(a)))
\end{aligned}
$$

Furthermore, we define the following notations:

- $m_1 \xrightarrow[w]{n} m_2$ if and only if $(m_1, n, m_2) \in R^w$. We say that in the wait context propagation phase marking m_1 enables node n and its firing results in m_2.
- $m \underset{w}{\rightarrow} m'$ if and only if $\exists n : m_1 \xrightarrow[w]{n} m_2$.
- $m \xrightarrow[w]{\tau} m'$ if and only if $\exists_{n_1,\ldots,n_q,m_1,\ldots,m_{q+1}} : \tau = n_1 n_2 \ldots n_q \in N* \ \wedge$ $m_1 = m \wedge m_{q+1} = m' \wedge m_1 \xrightarrow[w]{n_1} m_2, m_2 \xrightarrow[w]{n_2} \ldots \xrightarrow[w]{n_q} m_{q+1}$.
- $m \xrightarrow[w]{*} m'$ if and only if $\exists_\tau : m \xrightarrow[w]{\tau} m'$.

- $m \overset{max}{\underset{w}{\rightarrow}} m'$ if and only if $\exists_\tau : m \overset{\tau}{\underset{w}{\rightarrow}} m' \wedge \nexists_{m'' \neq m'} : m' \underset{w}{\rightarrow} m''$.
- $max_w : M_{EPC} \rightarrow M_{EPC}$ such that $max_w(m) = m'$ if and only if $m \overset{max}{\underset{w}{\rightarrow}} m'$. The existence of a unique $max_w(m)$ is the subject of Theorem 3.2 below.

Theorem 3.2 (Wait Context Propagation terminates)**.** *For an EPC and a given marking m, there exists a unique $max_w(m)$ which is determined in a finite number of propagation steps.*

Proof. Analogous proof as for Theorem 3.1. □

Phase 3: Transition Relation for Negative State Propagation

The transition rules for the various node types in this phase can be easily summarized in one transition relation: if all input arcs carry a negative token and all output arcs hold no negative or positive token, then consume all negative tokens on the input arcs and produce negative tokens on each output arc.

Definition 3.18 (Transition Relations for Negative State Propagation)**.** Let $EPC = (E, F, C, l, A)$ be a relaxed syntactically correct EPC, $N = E \cup F \cup C$ its set of nodes, and M_{EPC} its marking space. Then $R^{-1} \subseteq M_{EPC} \times N \times M_{EPC}$ is the transition relation for negative state propagation and $(m, n, m') \in R^{-1}$ if and only if:

$$
\begin{aligned}
&(\forall_{a \in n_{in}} : \sigma_m(a) = -1) \wedge \\
&(\forall_{a \in n_{out}} : \sigma_m(a) = 0) \wedge \\
&(\forall_{a \in n_{in}} : \sigma_{m'}(a) = 0) \wedge \\
&(\forall_{a \in n_{out}} : \sigma_{m'}(a) = -1) \wedge \\
&(\forall_{a \in A \setminus n_{out}} : \kappa_{m'}(a) = \kappa_m(a)) \wedge \\
&(\forall_{a \in n_{out}} : \kappa_{m'}(a) = dead) \wedge \\
&(\forall_{a \in A \setminus (n_{in} \cup n_{out})} : \sigma_{m'}(a) = \sigma_m(a))
\end{aligned}
$$

Furthermore, we define the following notations:

- $m_1 \overset{n}{\underset{-1}{\rightarrow}} m_2$ if and only if $(m_1, n, m_2) \in R^{-1}$. We say that in the negative state propagation phase marking m_1 enables node n and its firing results in m_2.

- $m \underset{-1}{\rightarrow} m'$ if and only if $\exists n : m_1 \xrightarrow[-1]{n} m_2$.
- $m \xrightarrow[-1]{\tau} m'$ if and only if $\exists_{n_1,...,n_q,m_1,...,m_{q+1}} : \tau = n_1 n_2 ... n_q \in N* \wedge$ $m_1 = m \wedge m_{q+1} = m' \wedge m_1 \xrightarrow[-1]{n_1} m_2, m_2 \xrightarrow[-1]{n_2} ... \xrightarrow[-1]{n_q} m_{q+1}$.
- $m \xrightarrow[-1]{*} m'$ if and only if $\exists_\tau : m \xrightarrow[-1]{\tau} m'$.
- $m \xrightarrow[-1]{max} m'$ if and only if $\exists_\tau : m \xrightarrow[-1]{\tau} m' \wedge \nexists_{m'' \neq m'} : m' \underset{-1}{\rightarrow} m''$.
- $max_{-1} : M_{EPC} \rightarrow M_{EPC}$ such that $max_{-1}(m) = m'$ if and only if $m \xrightarrow[-1]{max} m'$. The existence of a unique $max_{-1}(m)$ is discussed below in Theorem 3.3.

Theorem 3.3 (Negative State Propagation terminates)**.** *For an EPC and a given marking m, there exists a unique $max_{-1}(m)$ which is determined in a finite number of propagation steps.*

Proof. Regarding finiteness, by contradiction. Since an EPC is safe, i.e. there is at maximum one token per arc, it is a prerequisite for an infinite propagation that there is a cyclic structure in the process in which the negative token runs into an infinite loop. Due to the coherence property of an EPC, and the minimum number of one start and one end node (Definition 3.11), two cases of a cyclic path can be distinguished:

(i) cyclic path $a \hookrightarrow a$ with $\nexists e \in E_s : e \hookrightarrow a$: in this case the loop could potentially propagate a negative token infinitely, but it will never receive a token since there is no path from a start node into the cyclic path. Furthermore, relaxed syntactically correct EPCs do not contain such paths according to Definition 3.8.

(ii) cyclic path $a \hookrightarrow a$ with $\exists e \in E_s : e \hookrightarrow a$: In this case, there must be a join j on a cyclic path $a \hookrightarrow a$ such that there exists an arc (x, j) and there is no path $a \hookrightarrow x$. Therefore, a negative token could only be propagated infinitely on the path $a \hookrightarrow a$ if the join j would receive repeatedly ad infinitum negative tokens on the arc (x, j) in order to allow j to fire according to Definition 3.18. Since the number of tokens on arcs is limited to one, this is only possible if there is another cyclic path $b \hookrightarrow b$ that produces negative tokens ad infinitum on a split node s. Again, for this cyclic path $b \hookrightarrow b$, the two cases (i) and (ii) can be distinguished. Accordingly, there must be another cyclic path $c \hookrightarrow c$ that feeds the path with b, and so forth.

Since the existence of a cyclic path that propagates negative tokens infinitely depends on the existence of another such path, there is a contradiction. □

Regarding uniqueness we do not provide a formal proof here. Consider that there exist an original marking $m_0 \in M_{EPC}$ and two markings $m_{max1}, m_{max2} \in M_{EPC}$ such that $m_0 \overset{max}{\underset{-1}{\rightarrow}} m_{max1}$, $m_0 \overset{max}{\underset{-1}{\rightarrow}} m_{max2}$, and $m_{max1} \neq m_{max2}$. According to the transition relation, there are no transitions that could compete for tokens such as in non free-choice Petri nets, i.e. the firing of a transition cannot disable another one, and there are no alternative transitions for an enabled node. Furthermore, a context change of an arc has no impact on the applicability of a rule and no positive tokens are involved in firings. Therefore, m_{max1} and m_{max2} must either be equivalent or there must be a transition enabled in one of them such that the max property of it does not hold.

Phase 4: Transition Relation for Positive State Propagation

For OR-joins, we already described the concept of a negative upper corona in Section 3.4.4 on page 78. The firing of an OR-join consumes not only the negative tokens on its input arcs, but also the negative tokens on its negative upper corona. This way, no unnecessary negative tokens remain in the EPC.

Definition 3.19 (Dead Empty Path, Negative Upper Corona)**.** Let $EPC = (E, F, C, l, A)$ be a relaxed syntactically correct EPC, $N = E \cup F \cup C$ its set of nodes, and a marking $m \in M_{EPC}$. Then, we define the negative upper corona of a node $n \in N$ based on a dead empty path. A *dead empty path* $a \overset{d}{\underset{m}{\hookrightarrow}} b$ refers to a sequence of nodes $n_1, \ldots, n_k \in N$ with $a = n_1$ and $b = n_k$ such that for $(n_1, n_2) \in A : \sigma_m(n_1, n_2) = -1$ and $\forall i \in 2, \ldots, k-1$ holds: $(n_i, n_{i+1}) \in A \wedge \sigma_m(n_i, n_{i+1}) = 0 \wedge \kappa_m(n_i, n_{i+1}) = dead$. Then, the *negative upper corona* $\overset{-1}{\underset{m}{\hookrightarrow}} n = \{a \in A | a = (s,t) \wedge \sigma(a) = -1 \wedge t \overset{d}{\underset{m}{\hookrightarrow}} n\}$ refers to those arcs with a negative token whose target node t is a transitive predecessor of n and has a dead empty path to n in marking m.

The transition rules for the various node types can be easily summarized as follows: (1) for function, event, and AND-connector nodes, positive tokens on all input arcs are consumed and propagated to all output arcs, if all of them are empty. The input context

is set to dead and the output context to wait. (2) For XOR-connectors, one input token is consumed from one input arc and propagated to one of the output arcs if all of them are empty. The respective input arc is set to a dead context, as well as those output arcs that do not receive the token. The output arc with the positive token gets a wait context. (3) For OR-splits, the positive token is consumed from the input, and a combination of positive and negative tokens is produced at the output arcs such that at least one positive token is available. Furthermore, each output arc with a positive token gets a wait context while the others get a dead context. (4) OR-joins fire either if all input arcs are not empty and one of them has a positive token, or if there is no empty arc with a wait context and at least one positive token on the inputs. Then, all input tokens are consumed, plus potentially negative tokens on the negative upper corona, the input arcs are set to a dead context, and a positive token is produced on the output with a wait context.

Definition 3.20 (Transition Relation for Positive State Propagation)**.** Let $EPC = (E, F, C, l, A)$ be a relaxed syntactically correct EPC, $N = E \cup F \cup C$ its set of nodes, and M_{EPC} its marking space. Then $R^{+1} \subseteq M_{EPC} \times N \times M_{EPC}$ is the transition relation for positive state propagation and $(m, n, m') \in R^{+1}$ if and only if:

$$
\begin{aligned}
&((n \in F \cup E_{int} \cup C_{and}) \wedge \\
&(\forall_{a \in n_{in}} : \sigma_m(a) = +1) \wedge \\
&(\forall_{a \in n_{out}} : \sigma_m(a) = 0) \wedge \\
&(\forall_{a \in n_{in}} : \sigma_{m'}(a) = 0 \ \wedge \ \kappa_{m'}(a) = dead) \wedge \\
&(\forall_{a \in n_{out}} : \sigma_{m'}(a) = +1 \ \wedge \ \kappa_{m'}(a) = wait) \wedge \\
&(\forall_{a \in A \setminus (n_{in} \cup n_{out})} : \kappa_{m'}(a) = \kappa_m(a)) \wedge \\
&(\forall_{a \in A \setminus (n_{in} \cup n_{out})} : \sigma_{m'}(a) = \sigma_m(a))) \\
&\vee \\
&((n \in C_{xor}) \wedge \\
&(\exists_{a_1 \in n_{in}} : (\sigma_m(a_1) = +1 \wedge \sigma_{m'}(a_1) = 0 \wedge \\
&\qquad \kappa_m(a_1) = wait \ \wedge \ \kappa_{m'}(a_1) = dead) \wedge \\
&\qquad (\forall_{a \in n_{out}} : \sigma_m(a) = 0) \wedge \\
&\qquad (\exists_{X \wedge a_2 \in n_{out}} : X = \{a \in n_{in} \mid \sigma_m(a) = -1 \ \wedge \kappa_m(a) = dead\} \wedge \\
&\qquad\qquad (\sigma_{m'}(a_2) = +1 \ \wedge \ \kappa_{m'}(a_2) = wait) \wedge \\
&\qquad\qquad (\forall_{a \in A \setminus \{a_1, a_2\}} : \kappa_{m'}(a) = \kappa_m(a)) \wedge
\end{aligned}
$$

$$
\begin{aligned}
&(\forall_{a\in X} : \sigma_{m'}(a) = 0 \ \wedge \ \kappa_{m'}(a) = \kappa_m(a)) \wedge \\
&(\forall_{a\in A\setminus(X\cup\{a_1,a_2\})} : \sigma_{m'}(a) = \sigma_m(a)))))\\
\vee\\
&((n \in S_{or}) \wedge \\
&(\forall_{a\in n_{in}} : \sigma_m(a) = +1) \wedge \\
&(\forall_{a\in n_{out}} : \sigma_m(a) = 0) \wedge \\
&(\forall_{a\in n_{in}} : \sigma_{m'}(a) = 0 \ \wedge \kappa_{m'}(a) = dead) \wedge \\
&(\exists_{X\neq\emptyset} : X = \{a \in n_{out} \mid \sigma_{m'}(a) = +1 \ \wedge \kappa_{m'}(a) = wait\} \wedge \\
&\qquad (\forall_{a\in n_{out}\setminus X} : \sigma_{m'}(a) = -1 \ \wedge \kappa_{m'}(a) = dead) \wedge \\
&\qquad (\forall_{a\in A\setminus(n_{in}\cup n_{out})} : \kappa_{m'}(a) = \kappa_m(a) \wedge \sigma_{m'}(a) = \sigma_m(a)))\\
\vee\\
&((n \in J_{or}) \wedge \\
&(\exists_{X\neq\emptyset} : X = \{a \in n_{in} \mid \sigma_m(a) = +1 \ \wedge \kappa_m(a) = wait\}) \wedge \\
&(\exists_{Y} : Y = \{a \in n_{in} \mid \sigma_m(a) = -1 \ \wedge \kappa_m(a) = dead\}) \wedge \\
&(\exists_{Z} : Z = \{a \in n_{in} \mid \sigma_m(a) = 0 \ \wedge \kappa_m(a) = dead\}) \wedge \\
&(X \cup Y \cup Z = n_{in}) \wedge \\
&(\forall_{a\in n_{out}} : \sigma_m(a) = 0) \wedge \\
&(\forall_{a\in n_{in}} : \sigma_{m'}(a) = 0 \wedge \kappa_{m'}(a) = dead)) \wedge \\
&(\forall_{a\in n_{out}} : \sigma_{m'}(a) = +1 \wedge \kappa_{m'}(a) = wait) \wedge \\
&(\exists_{U\subset A} : U = \overset{-1}{\underset{m}{\hookrightarrow}} n \wedge \\
&\qquad (\forall_{a\in U} : \sigma_{m'}(a) = 0 \wedge \kappa_{m'}(a) = \kappa_m(a)) \wedge \\
&\qquad (\forall_{a\in A\setminus(U\cup n_{in}\cup n_{out})} : \sigma_{m'}(a) = \sigma_m(a) \wedge \kappa_{m'}(a) = \kappa_m(a)))).
\end{aligned}
$$

Furthermore, we define the following notations:

- $m_1 \overset{n}{\underset{+1}{\rightarrow}} m_2$ if and only if $(m_1, n, m_2) \in R^{+1}$. We say that in the positive state propagation phase marking m_1 enables node n and its firing results in m_2.
- $m \underset{+1}{\rightarrow} m'$ if and only if $\exists n : m_1 \overset{n}{\underset{+1}{\rightarrow}} m_2$.
- $m \overset{\tau}{\underset{+1}{\rightarrow}} m'$ if and only if $\exists_{n_1,\ldots,n_q,m_1,\ldots,m_{q+1}} : \tau = n_1 n_2 \ldots n_q \in N* \ \wedge$
  $m_1 = m \wedge m_{q+1} = m' \wedge m_1 \overset{n_1}{\underset{+1}{\rightarrow}} m_2, m_2 \overset{n_2}{\underset{+1}{\rightarrow}} \ldots \overset{n_q}{\underset{+1}{\rightarrow}} m_{q+1}$.

- $m \xrightarrow[+1]{*} m'$ if and only if $\exists_\tau : m \xrightarrow[+1]{\tau} m'$.

Since the transition relation covers several marking changes that are not interesting for an observer of the process, we define the reachability graph RG of an EPC in the following section. It includes only transitions of the positive state propagation phase.

Calculating the Reachability Graph for EPCs

In this section, we define the reachability graph of an EPC and present an algorithm to calculate it. First we formalize the concept of reachability related to an EPC.

Definition 3.21 (Reachability related to an EPC)**.** Let $EPC = (E, F, C, l, A)$ be a relaxed syntactically correct EPC, $N = E \cup F \cup C$ its set of nodes, and M_{EPC} its marking space. Then, a marking $m' \in M_{EPC}$ is called reachable from another marking m if and only if $\exists n \in N \ \wedge \ m_1, m_2, m_3 \in M_{EPC} : max_d(m) = m_1 \wedge max_w(m_1) = m_2 \wedge max_{-1}(m_2) = m_3 \wedge m_3 \xrightarrow[+1]{n} m'$. Furthermore, we define the following notations:

- $m \xrightarrow{n} m'$ if and only if m' is reachable from m.
- $m \rightarrow m' \Leftrightarrow \exists n \in N : m \xrightarrow{n} m'$.
- $m \xrightarrow{\tau} m'$ if and only if $\exists_{n_1,...,n_q,m_1,...,m_{q+1}} : \tau = n_1 n_2 ... n_q \in N* \ \wedge$ $m_1 = m \wedge m_{q+1} = m' \wedge m_1 \xrightarrow{n_1} m_2, m_2 \xrightarrow{n_2} ... \xrightarrow{n_q} m_{q+1}$.
- $m_1 \xrightarrow{*} m_q \Leftrightarrow \exists \tau : m_1 \xrightarrow{\tau} m_q$.

Definition 3.22 (Reachability Graph of an EPC)**.** Let $EPC = (E, F, C, l, A)$ be a relaxed syntactically correct EPC, $N = E \cup F \cup C$ its set of nodes, and M_{EPC} its marking space. Then, the reachability graph $RG \subseteq M_{EPC} \times N \times M_{EPC}$ of an EPC contains the following nodes and transitions:

(i) $\forall m \in I_{EPC} : m \in RG$.

(ii) $(m, n, m') \in RG$ if and only if $m \xrightarrow{n} m'$.

The calculation of RG requires an EPC as input and a set of initial markings $I \subseteq I_{EPC}$. For several EPCs from practice, such a set of initial markings will not be available.

In this case, one can easily calculate the set of all possible initial markings. Algorithm 1 uses an object-oriented pseudo code notation to define the calculation. In particular, we assume that RG is an instance of the class $ReachabilityGraph$, $propagated$ an instances of class Set, and $toBePropagated$ an instance of class $Stack$ that provides the methods $pop()$ and $push()$. Furthermore, $currentMarking$, $oldMarking$, and $newMarking$ are instances of class $Marking$ that provides the methods $clone()$ to return a new, but equivalent marking, $propagateDeadContext(EPC)$, $propagateWaitContext(EPC)$, and $propagateNegativeTokens(EPC)$ to change the marking according to the transitions of the respective phase, i.e. to determine max_d, max_w, and max_{-1} of the current marking. Finally, $propagatePositiveTokens(EPC)$ returns a set of (node,marking) pairs including the node that can fire and the marking that is reached after the firing.

In lines 1-3, the sets RG and $propagated$ are initialized with the empty set, and the stack $toBePropagated$ is filled with all initial markings of the set I_{EPC}. The while loop between lines 4-18 calculates new markings for the marking that is on top of the stack $toBePropagated$. In particular, $currentMarking$ receives the top marking from the stack (line 5), and it is cloned into the $oldMarking$ object (line 6). In lines 7-9, the propagations of dead and wait context and of negative tokens are applied on $currentMarking$. Then, in line 10, the pairs of nodes and new markings that can be reached from the old marking are stored in the set $nodeNewMarking$. After that, the old marking is added to the $propagated$ set (line 11). In lines 12-17, for each pair of node and new marking a new transition $(oldMarking, node, newMarking)$ is added to RG. If a new marking has not yet been propagated, it is pushed on top of the $toBePropagated$ stack (lines 14-16). Using a stack, the reachability graph is calculated in a depth-first manner. Finally, in line 19 RG is returned.

3.4.6 Tool Support for the Novel EPC Semantics

Based on the previous algorithm, we have implemented the novel EPC semantics as a conversion plug-in for the *ProM* (Process Mining) framework [DMV+05, VDMA06, BHK+06]. ProM was originally developed as a tool for *process mining*, which is a domain that aims at extracting information from event logs to capture the business process as

Algorithm 1 Pseudo code for calculating the reachability graph of an EPC

Require: $EPC = (E, F, C, l, A), I \subseteq M$
1: $RG \leftarrow \emptyset$
2: $toBePropagated \leftarrow I_{EPC}$
3: $propagated \leftarrow \emptyset$
4: **while** $toBePropagated \neq \emptyset$ **do**
5: $currentMarking \leftarrow toBePropagated.pop()$
6: $oldMarking \leftarrow currentMarking.clone()$
7: $currentMarking.propagateDeadContext(EPC)$
8: $currentMarking.propagateWaitContext(EPC)$
9: $currentMarking.propagateNegativeTokens(EPC)$
10: $nodeNewMarking \leftarrow currentMarking.propagatePositiveTokens(EPC)$
11: $propagated.add(oldMarking)$
12: **for all** $(node, newMarking) \in nodeNewMarkings$ **do**
13: $RG.add(oldMarking, node, newMarking)$
14: **if** $newMarking \notin propagated$ **then**
15: $toBePropagated.push(newMarking)$
16: **end if**
17: **end for**
18: **end while**
19: **return** RG

it is being executed (cf. e.g. [ADH+03, AWM04, CW98, GCC+04, Her00]). In the meantime, the functionality of ProM was extended to include other types of analysis, model conversions, model comparison, etc. This was enabled by the plug-able architecture of ProM, that allows to add new functionality without changing the framework itself, and the fact that ProM supports multiple modeling languages. Since ProM can interact with a variety of existing systems, e.g., *workflow management systems* such as Staffware, Oracle BPEL, Eastman Workflow, WebSphere, InConcert, FLOWer, Caramba, and YAWL, *simulation tools* such as ARIS, EPC Tools, Yasper, and CPN Tools, *ERP systems* like PeopleSoft and SAP, *analysis tools* such as AGNA, NetMiner, Viscovery, AlphaMiner, and ARIS PPM (cf. [BHK+06]), the plug-in for the new EPC semantics can easily be used for the analysis of existing models. Currently, there are more than 150 plug-ins in release 4.1. ProM basically supports five kinds of plug-ins:

Mining plug-ins to take a log and produce a model,

Import plug-ins to import a model from file, and possibly use a log to identify the relevant objects in the model,

Export plug-ins to export a model to file,

Conversion plug-ins to convert one model into another, and

Analysis plug-ins to analyze a model, potentially in combination with a log.

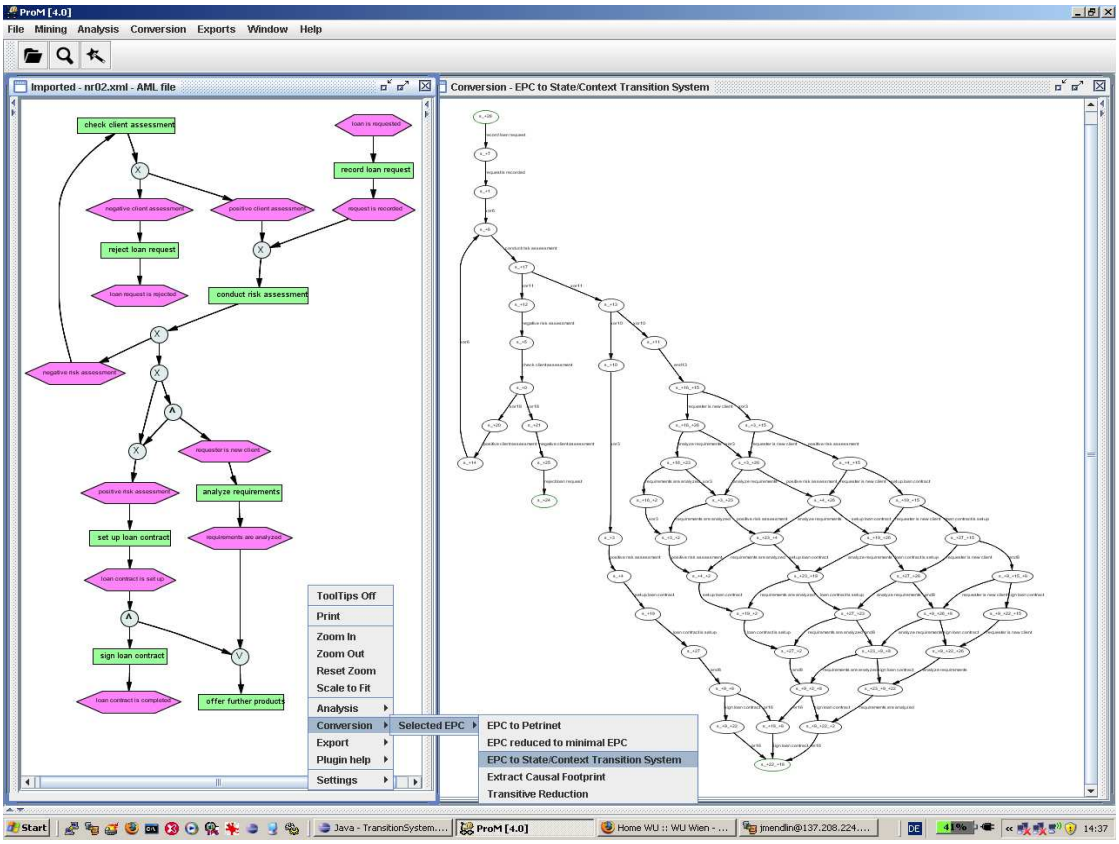


Figure 3.20: Calculating the reachability graph in ProM

The conversion plug-in maps an EPC to the transition systems package (cf. [ARD+06, RGA+06]) that was developed for an implementation of the incremental workflow mining approach by *Kindler, Rubin, and Schäfer* [KRS05, KRS06a, KRS06b]. Figure 3.20 illustrates how the conversion plug-in works. First, one has to load an EPC business process model into ProM, for instance, by using the import plug-in for the ARIS XML

format [IDS03b] or for the EPC Markup Language [MN06]. In the figure, the EPC example model for a loan request process that we introduced in the beginning of this chapter is loaded. Since ProM generates a new layout automatically, the model looks different compared to the previous figure. Once the EPC is displayed in ProM, one can click on it, trigger the conversion plug-in "EPC to State/Context Transition System", and the reachability graph is calculated and shown in a new ProM window. The dense network of states and transitions on the right-hand side stems from the concurrent execution, if there is both a positive risk assessment for the loan request and the requester is a new customer. There are two markings that do not serve as a source for another transition in case if the request is rejected or accepted. Both these markings are displayed with a green border since they are proper final markings. If they were deadlocks, they would be drawn with a red border.

One of the nice features of the transition system package is that it provides an export to the file format of Petrify. *Petrify* is a software tool developed by *Cortadella, Kishinevsky, Lavagno, and Yakovlev* [CKLY98, Cor98] that can not only generate the state space for a Petri net, but also a Petri net from a transition system. The concepts of this Petri net synthesis builds on the theory of regions by *Ehrenfeucht and Rozenberg* [ER89, BD98]. Running Petrify with the reachability graph of the Loan Request example EPC of Figure 3.1 generates a free-choice Petri net as shown in Figure 3.21. It is interesting to see how the OR-join $or16$ is treated in the Petri net synthesis. It requires a token at each of the two input places before it can fire. If both the *positive risk assessment* and the *requester is new client* branch are executed, the OR-join synchronizes these paths via its two input places. If only the *positive risk assessment* branch is executed, the required tokens are produced by $xor3$. The decision point $xor11$ is the same as in the EPC model. Furthermore, it can be seen that each alternative of an XOR-split becomes a transition of its own (see $xor10$ and $xor10.\_1$ or $xor11$ and $xor11.\_1$) while the AND-split $and13$ remains one transition in the Petri net. The generation of a reachability graph for an EPC and the synthesis of a Petri net could be an important step to bring EPCs and Petri nets closer together. In particular, such a procedure could be a way to get rid of OR-joins for a Petri net implementation that has been modelled with EPCs in the design phase.

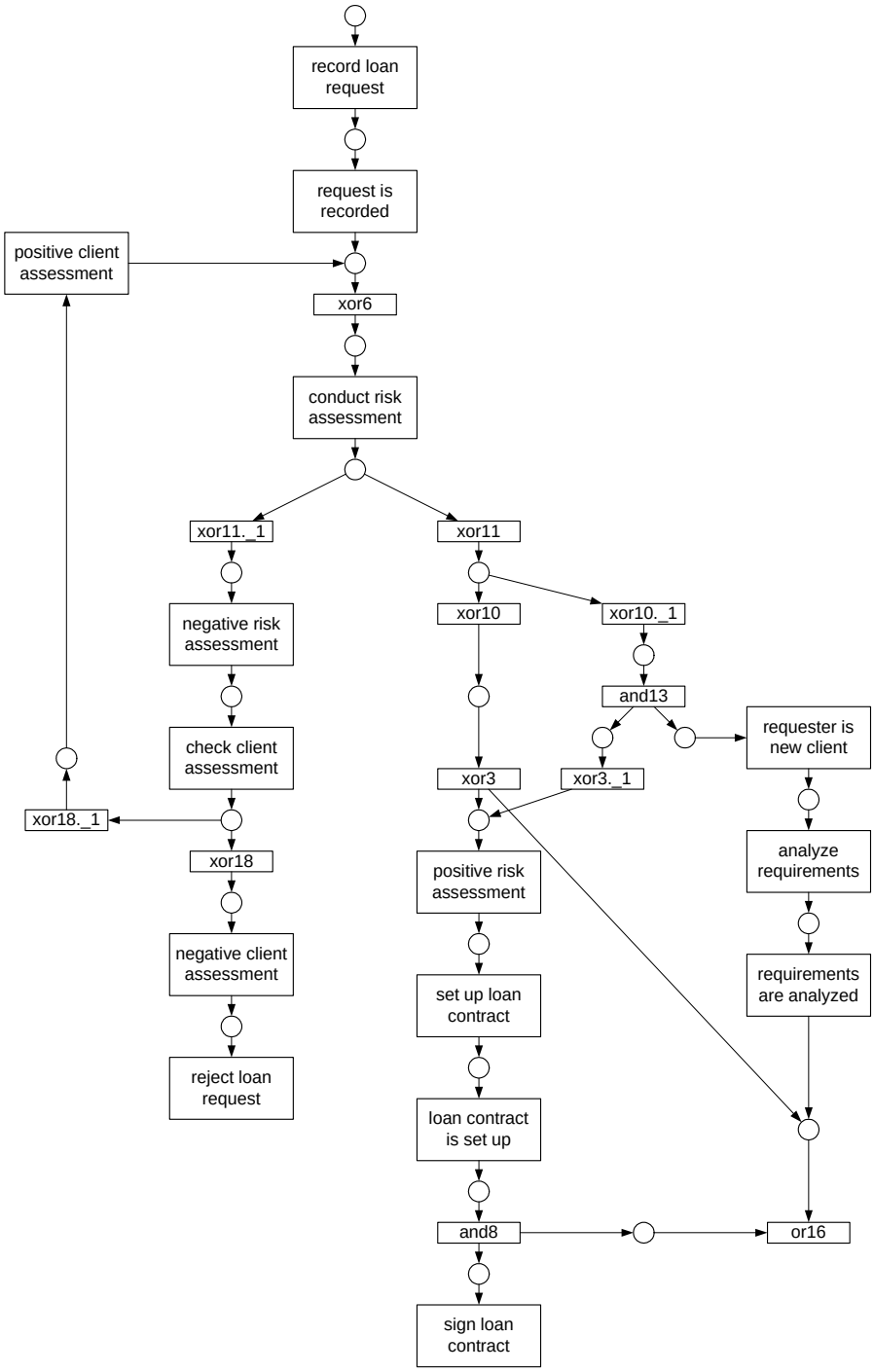


Figure 3.21: A Petri net that is bisimilar to the Loan Request EPC

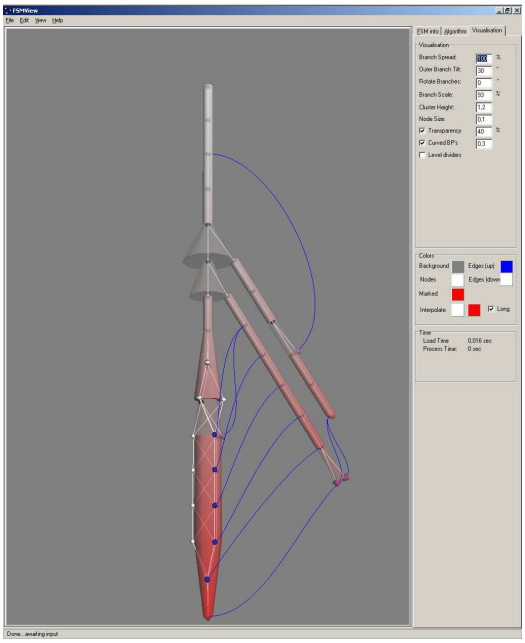


Figure 3.22: A visualization of the state space of the Loan Request Petri net

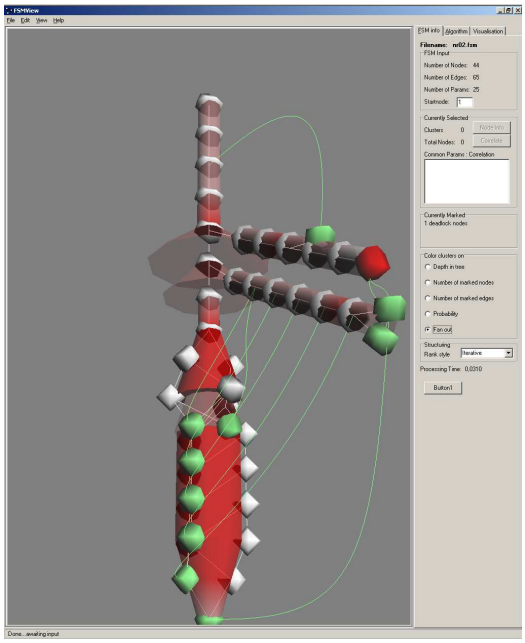


Figure 3.23: Another visualization of the Loan Request state space

Another useful application related to the ProM plug-in is the possibility to export to the FSM format via the Petri net analysis plug-in in ProM. This format can be loaded into the visualization tool FSMTool by *Groote and Van Ham* [HWW02, GH03, GH06]. FSMTool provides sophisticated interactive and customizable visualization of large state transition systems. The general visualization principle of FSMTool is to project the state space on levels of a backbone in such a way that structural symmetry can easily be seen. The Figures 3.22 and 3.23 visualize the state space of the Loan Request Petri net that was generated by Petrify as a three-dimensional backbone. The two decision points of this process are represented as cones in the upper part of the backbone. Each of these decision points splits off a new branch of execution that is visualized as a separate arm. On the first arm for negative risk assessment, there is a green line in Figure 3.22 (in Figure 3.23 it is blue) that represents an iteration of the loop. The other green lines highlight the activation of a node that is closer to the start node than the node that had control before. The thick pillar of the backbone represents the parallel execution after the AND-split. Overall, the FSMTool is a useful addition to the ProM plug-ins for understanding the complexity of

the state space. Still, certain information about function labels is not present and there is no direct connection to the process model.

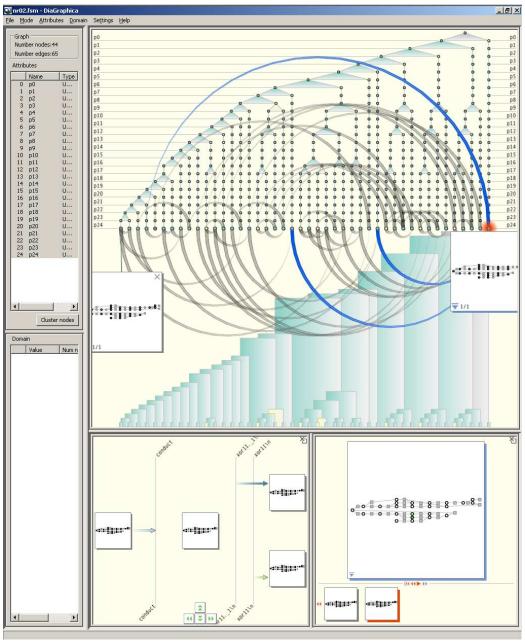


Figure 3.24: Visualization of the Petri net and the state space in DiaGraphica

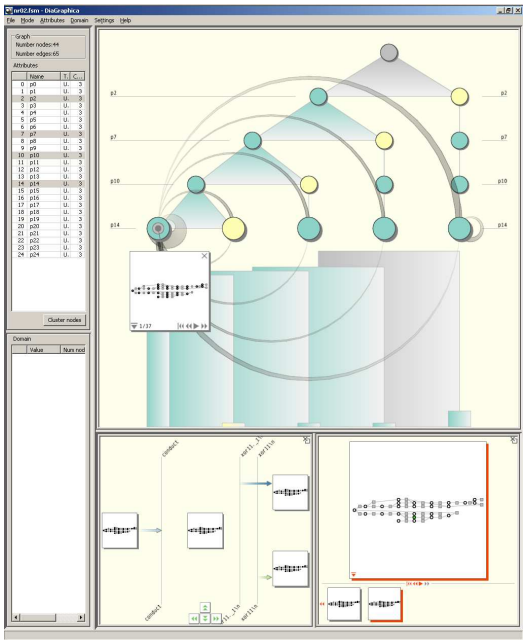


Figure 3.25: Clustering of places for the same state space in DiaGraphica

This shortcoming is the motivation of the work by *Verbeek, Pretorius, Van der Aalst, and Van Wijk* [VPAJ07] on a two-dimensional projection of state spaces as an extension to the Diagraphica tool of *Pretorius and Van Wijk* [PW05, PW06a]. Diagraphica can also load FSM files and in addition the diagram of a Petri net. Figure 3.24 shows that DiaGraphica uses an attribute clustering technique where, in this case, the attributes are related to the places of the Petri net. As Figure 3.25 shows, there may be multiple places in a cluster depending on the selections of the user. Transitions are represented as arcs. This figure permits an interesting observation. Below the diagonal line of yellow clusters the clustering hierarchy does not branch anymore. This means that for the selected places, only one can be marked at the same time (cf. [VPAJ07, p.16]). Further interpretations of different clustering patterns are discussed in [VPAJ07].

Based on the implementation of the reachability graph calculation in ProM, we can relate the novel EPC semantics to several other tools and approaches for analysis, syn-

thesis, and visualization of process models and state spaces. This way, researchers can easily benefit from the EPC semantics and analyze its relationship to other formalisms.

3.5 EPCs and other Process Modeling Languages

In this section, we provide a comparison of EPCs with other business process modeling languages. The selection includes Workflow nets [Aal97], UML Activity Diagrams (UML AD) [OMG04], BPMN [OMG06], and YAWL [AH05], and is meant to illustrate differences and commonalities without going into mapping details. We first discuss whether these other process modeling languages offer elements similar to the different EPC connectors. After that, we utilize the workflow patterns documented in [AHKB03] to compare the languages. BPEL [CGK+02, ACD+03, AAB+05], which is also receiving increasing attention as a standard, is not included here since it addresses the execution rather than the conceptual modeling of processes. For further details on the relationship between EPCs and BPEL, refer to [MZ05a, ZM05, MZ05b, MLZ05, MLZ06b, MLZ06a]. For a workflow pattern analysis of BPEL, see [WADH03]. Furthermore, the XPDL standard [Wor02, Wor05] has also gained some support in the industry for the definition of executable workflow process. A workflow pattern analysis of XPDL is reported in [Aal03]. Other approaches for comparing process modeling languages are reported in [SAJ+02, RG02, BKKR03, Mue04, LK06].

3.5.1 Comparison based on Routing Elements

The six different connectors of EPCs, i.e., XOR-split and XOR-join, AND-split and AND-join, OR-split and OR-join, provide the means to model complex routing and ordering between activities of a business process. Table 3.4 takes these routing elements as a benchmark to compare EPCs with other business process modeling languages. It shows that the behavioral semantics of XOR-connectors and AND-connectors, as well as OR-split connectors, can be represented in all the considered languages. In *Workflow nets* XOR-connectors and AND-connectors are captured by places and transitions with multiple input and output arcs, respectively. OR-split behavior can be specified as

a complex subnet that determines each possible combination of inputs. OR-join behavior cannot be modelled directly, but a relaxed soundness analysis is possible. In *UML AD* the XOR-split maps to a Decision, the XOR-join to a Merge, the AND-split to a Fork, the AND-Join to a Join, and the OR-split to a Fork with guards on its output arcs. OR-joins cannot be represented in UML AD directly. In *BPMN*, routing elements are called gateways. Basically, each EPC connector can be transformed to a respective gateway. In *YAWL*, there are also similar splits and joins matching the behavior of the EPC connectors.

Table 3.4: EPC routing elements and equivalent elements in other business process modeling languages

EPC	Workflow nets	UML AD	BPMN	YAWL
XOR-split	multi-out place	Decision	XOR-gateway	XOR-split task
XOR-join	multi-in place	Merge	XOR-gateway	XOR-join task
AND-split	multi-out transition	Fork	AND-gateway	AND-split task
AND-join	multi-in transition	Join	AND-gateway	AND-join task
OR-split	complex subnet	Fork	OR-gateway	OR-split task
OR-join	-	-	OR-gateway	OR-join task

3.5.2 Comparison based on Workflow Patterns

Motivated by the heterogeneity of workflow languages and products, *Van der Aalst, Ter Hofstede, Kiepuszewski, and Barros* have gathered a set of 20 workflow patterns [AHKB03]. These patterns can be utilized to clarify semantics or to serve as a benchmark. Table 3.5 illustrates the result of several workflow pattern analyses of EPCs [MNN05a], Workflow nets [AH05], UML AD [WAD+05], BPMN [WAD+06], and YAWL [AH05]. It can be seen that EPCs support the basic control flow patterns, multiple choice, and synchronizing merge. These patterns can be directly represented with the different EPC connectors. Furthermore, EPCs permit arbitrary cycles and offer implicit termination. Multiple instances with apriori design time knowledge can be modelled by an AND-block, with as many instances as required of the same activity in parallel. The yEPC extension provides support for all patterns.

In contrast to EPCs, *Workflow nets* support the state-based patterns, but perform weak

Table 3.5: Workflow pattern support of EPCs and other business process modeling languages

Workflow Pattern	EPC	Wf. nets	UML AD	BPMN	YAWL
Basic Control Flow Patterns					
1. Sequence	+	+	+	+	+
2. Parallel Split	+	+	+	+	+
3. Synchronization	+	+	+	+	+
4. Exclusive Choice	+	+	+	+	+
5. Simple Merge	+	+	+	+	+
Advanced Branching and Synchronization Patterns					
6. Multiple Choice	+	+	+	+	+
7. Synchronizing Merge	+	-	-	+/-	+
8. Multi Merge	-	+	+	+	+
9. Discriminator	-	-	+	+/-	+
Structural Patterns					
10. Arbitrary Cycles	+	+	+	+	+
11. Implicit Termination	+	-	+	+	-
Patterns involving Multiple Instantiation (MI)					
12. MI without Synchronization	-	+	+	+	+
13. MI with apriori Design Time Knowledge	+	+	+	+	+
14. MI with apriori Runtime Knowledge	-	-	+	+	+
15. MI without apriori Runtime Knowledge	-	-	-	-	+
State-based Patterns					
16. Deferred Choice	-	+	+	+	+
17. Interl. Parallel Routing	-	+	-	+/-	+
18. Milestone	-	+	-	-	+
Cancellation Patterns					
19. Cancel Activity	-	+/-	+	+	+
20. Cancel Case	-	-	+	+	+

when it comes to advanced branching and synchronization patterns. *UML AD* cover several patterns missing only the synchronizing merge, multiple instances without apriori runtime knowledge, and two state-based patterns. *BPMN* performs even better since it supports the synchronizing merge, but only in a structured block. As *YAWL* was defined to provide a straight-forward support for the workflow patterns, it is no surprise that it has the best score. The implicit termination pattern is not supported in order to force the designer to make the completion condition explicit. The comparison reveals that the patterns supported by EPCs are, in most cases, also supported by the other languages. Because of this large overlap, several of the findings that are elaborated throughout the remainder of this thesis can be more or less directly applied to the other languages, in particular to YAWL.

3.6 Summary

In this chapter, we gathered state of the art work on EPCs. Building on the foundations of prior work, we established a novel syntax definition and a novel semantics definition for EPCs. Our semantics is based on transition relations that defines both state changes and context changes. Furthermore, we presented an algorithm to calculate the reachability graph of an EPC, based on the transition relation and a respective implementation as a plug-in for ProM. The major motivations for this novel semantics are, firstly, semantic gaps and, secondly, non-intuitive behavior of existing formalizations. The comparison to other business process modeling languages revealed that EPCs share their routing elements with several other process modeling languages. Therefore, the findings that are elaborated throughout the remainder of this thesis can be adapted to these languages in future research.

Chapter 4

Verification of EPC Soundness

The aim of this thesis is to evaluate the power of metrics for predicting errors in business process models. In order to do so, we need to establish a clear and unambiguous understanding of which EPC business process model is correct and how we can verify it. This chapter presents verification techniques that can be applied to identify errors in EPCs. In this context, we focus on reachability graph analysis and reduction rules. Other verification techniques such as calculating invariants (see e.g. [Mur89, VA06]), reasoning (see e.g. [Pnu77, DMA06]), or model integration (see e.g. [SM06]) are not considered.

In the first part of this chapter (Section 4.1), we motivate and define a notion of soundness for EPC business process models and show in Section 4.2 how the analysis of the reachability graph can be applied to verify soundness of an EPC given the semantics introduced in the previous chapter. Furthermore, we present an implementation of the analysis as an extension of the EPC to Transition System plug-in for ProM. Since this verification approach suffers from the "state explosion" problem, we turn to an optimization based on a set of reduction rules (Section 4.3). For the reduction rules, approach we present the implementation as a batch program called *xoEPC* and show the results of reducing the SAP reference model. Finally, Section 4.4 summarizes the chapter.

4.1 Soundness of EPCs

This section discusses existing correctness criteria for business process models in Section 4.1.1, and proposes a novel soundness notion that directly relates to multiple start and end events of EPCs in Section 4.1.2. After that, Section 4.2 presents how the reachability graph can be utilized for the verification of EPC soundness.

4.1.1 Correctness criteria for business process models

Soundness is an important and prominent correctness criterion for business process models and was first introduced by *Van der Aalst* in [Aal97]. The original soundness property is defined for a Workflow net, a Petri net with one source and one sink, and requires that (i) for every state reachable from the source, there exists a firing sequence to the sink (option to complete); (ii) the state with a token in the sink is the only state reachable from the initial state with at least one token in it (proper completion); and (iii) there are no dead transitions [Aal97]. Furthermore, *Van der Aalst* shows that soundness of a Workflow net is equivalent to liveness and boundedness of the corresponding short-circuited Petri net.[1] Therefore, several liveness and boundedness analysis techniques are directly applicable to the verification of soundness. The soundness property can be verified with Petri net analysis tools such as Woflan [VA00, VBA01, Ver04].

The soundness property of workflow models has stimulated the specification of several soundness derivatives, basically because some soundness aspects proved to be too restrictive in certain application domains. *Dehnert and Rittgen* argue that business processes are often conceptually modelled in such a way that only the desired behavior results in a proper completion. Since such models are not used for workflow execution, non-normative behavior is resolved by the people working in the process in a cooperative and ad-hoc fashion. Accordingly, they define a process to be *relaxed sound* if every tran-

[1] "A Petri net is said to be *k-bounded* or simply *bounded* if the number of tokens in each place does not exceed a finite number k for any marking that is reachable from the initial marking" [Mur89, p.547]. "A Petri net is said to be *live* if, no matter what marking has been reached from the initial marking, it is possible to ultimately fire *any* transition of the net by progressing through some further firing sequence" [Mur89, p.548].

sition in a Petri net representation of the process model is included in at least one proper execution sequence [DR01]. As already mentioned in Section 3.4.3, relaxed soundness can be used to analyze EPCs: if OR-joins are mapped to a Petri net block (see [Deh02]), the Petri net state space is larger than the actual state space with synchronization. Based on the relaxed soundness criterion, it is possible to check whether a join should synchronize (cf. [DA04]).

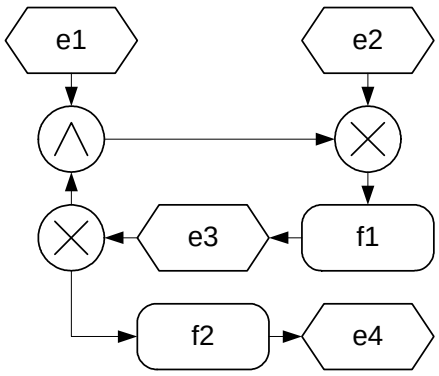


Figure 4.1: A relaxed sound EPC with structural problems

Figure 4.1 illustrates a subtle implication of the relaxed soundness definition. Consider an initial marking that includes both start arcs after $e1$ and $e2$. Entering the loop at the XOR-join, the right token can be propagated via the XOR-split to synchronize with the left token at the AND-join. If after that the loop is exited at the XOR-split, the process can complete properly. Since this execution sequence covers all nodes, the model is relaxed sound. Still, there is a structural problem because the loop can never be executed another time without running into a deadlock. Furthermore, the right token must never leave the loop without synchronizing with the left token at the AND-join. Therefore, the relaxed soundness criterion is in some cases too weak. The fact, that relaxed soundness process models can still include livelocks and deadlocks, is a motivation for *Puhlmann and Weske* to define a notion of *lazy soundness* [PW06b]. A lazy sound process is deadlock and livelock free as long as the final node has not been reached. Therefore, clean-up activities, such as cancelling parts of the process, are still permitted. Since such cleaning-up is needed for some of the workflow patterns, the authors also reject *weak soundness* defined by *Martens* [Mar03], which does not provide this feature. Still, both weak and

lazy soundness allow dead activities.

Soundness was also extended towards *k-soundness* in order to study processes with shared resources [BP98]. In this context, $k > 1$ refers to the number of tokens that are allowed on the initial and final place. Related to that, *generalized soundness* [HSV04] means that a process is k-sound for all $k > 1$; and *structural soundness* is fulfilled if there exists a $k > 1$ such that the process is k-sound. Both generalized and structural soundness are decidable (see [TM05, HSV04]) and a verification approach for generalized soundness is reported in [HOSV06]. Relationships between the different soundness notions are discussed in [Too04].

Beyond the soundness property, *structuredness* (or well-structuredness) is also discussed as a correctness criterion (see e.g. [Aal98]). In essence, a structured process can be constructed by nesting simple building blocks like split and join of the same connector type. We used such a structured OR-block in Section 3.4.2 to illustrate how refinement can affect the behavior in some EPC formalizations. Structuredness of a process model guarantees soundness if the model is live (see e.g. [DZ05]). Some process modeling languages, like BPEL and several workflow systems (cf. [Kie03]), enforce the definition of a structured model by imposing syntactical restrictions[2] in order to provide correctness by design (see [LR00, CGK+02, ACD+03, AAB+05]. Moreover, finding a structured model with equivalent behavior to an originally unstructured model is also used as a verification technique (see e.g. [KHB00, AJL05, AL05, LK05, ZHB+06, HFKV06]). Yet, structuredness as a correctness criterion has been criticized for being too strict (see e.g. [DZ05]) since some sound process models are discarded right from the start. Furthermore, nesting of structured blocks neither meets the way people comprehend processes nor does every process easily fit into this scheme. Therefore, structuredness should rather be regarded as a general guideline from which one can deviate if necessary.

Figure 4.2 summarizes the correctness criteria of soundness, relaxed soundness, structuredness, and their relations, cf. [DZ05]. Furthermore, it relates the properties to the Petri net classes of free-choice nets and state machines.[3] In particular, it highlights that

[2] BPEL relaxes structured modeling by allowing synchronization links between parallel activities.

[3] A *free-choice net* is a Petri net in which a place that is in the preset of multiple transitions is the only place in all these presets (see [DE95]). Therefore, the choice is "free" in a sense that it can be made

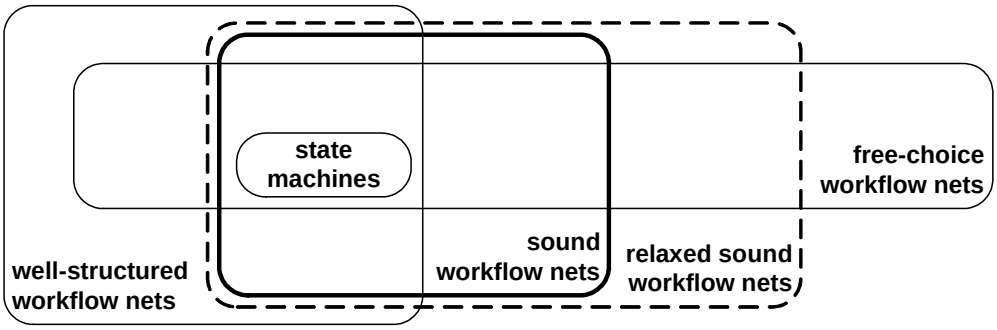


Figure 4.2: Relations between different Petri net-properties (see [DZ05, p.389])

a sound process model is also relaxed sound, and that a model that is both relaxed sound and structured is also sound. In the following, we aim to analyze EPCs with respect to a strict correctness criterion that guarantees that the models can be utilized in subsequent phases of the BPM life cycle as defined in Section 2.2. While the soundness definition would be a candidate, it is not directly applicable for EPCs: workflow nets have one unique start and one unique end node, but EPCs may have multiple start and end events. Accordingly, we will have to consider several initial and several final markings related to an EPC-specific soundness criterion.

4.1.2 Definition of EPC Soundness

The soundness definition for workflow nets cannot be used directly for EPCs since they may have multiple start and end events. Based on the definitions of the initial and final states of an EPC, we define soundness of an EPC analogously to soundness of Workflow nets [Aal97]. According to *Rump* [Rum99], there must be a set of initial markings for an EPC such that there exists at least one initial marking in which a particular start arc holds a positive token. Therefore, we demand in the soundness definition that there exists such a set of initial markings, and that for each initial marking in it proper completion is guaranteed. Analogously, we demand that there exists a set of final markings reachable

without considering other places. A *state machine* is a subclass of free-choice nets in which each transition has exactly one pre- and one postcondition (cf. e.g. [Des05]). Therefore, there is no concurrency in a state machine.

from some of these initial markings such that there exists at least one final marking in which a particular end arc holds a positive token. If that is fulfilled, every arc contributes to properly completing behavior of the EPC. The requirement that the EPC has to be relaxed syntactically correct excludes those pathological EPCs for which no semantics can be determined, for example, if there are multiple input arcs of a function, or if there are loops without an entry or exit connector.

Definition 4.1 (Soundness of an EPC)**.** Let $EPC = (E, F, C, l, A)$ be a relaxed syntactically correct EPC, $N = E \cup F \cup C$ its set of nodes, M_{EPC} its marking space, and I_{EPC} and O_{EPC} the set of possible initial and final markings. An EPC is sound if there exists a non-empty set of initial markings $I \subseteq I_{EPC}$ and a set of final markings $O \subseteq O_{EPC}$ such that:

(i) For each start-arc a_s there exists an initial marking $i \in I$ where the arc (and hence the corresponding start event) holds a positive token. Formally:
$\forall a_s \in A_s : \exists i \in I : \sigma_i(a_s) = +1$

(ii) For every marking m reachable from an initial state $i \in I$, there exists a firing sequence leading from marking m to a final marking $o \in O$. Formally:
$\forall i \in I : \forall m \in M : \ (i \xrightarrow{*} m) \Rightarrow \exists o \in O \ (m \xrightarrow{*} o)$

(iii) The final markings $o \in O$ are the only markings reachable from a marking $i \in I$ such that there is no node that can fire. Formally:
$\forall m \in M : ((i \rightarrow m) \wedge \nexists m'(m \rightarrow m')) \Rightarrow m \in O$

This soundness definition deserves some comments with respect to dead nodes, livelocks, and contact situations. The original soundness definition requires that a workflow net must not include *dead transitions* (property iii). In Definition 4.1 it is not explicitly demanded that there are no dead arcs. Still, this property is implicitly granted due to the fact that the EPC is relaxed syntactically correct and that all decisions of an EPC are free-choice. Together with EPC soundness (i), it follows that an arc can either be reached by some token from a start arc that carries a positive token in some initial marking, or there must be a deadlock on the path between the start arcs and the respective arc. If the latter is the case, the EPC is not sound because (iii) is violated. We summarize this property in the following observation without a proof.

Observation 4.1 (No dead nodes in sound EPCs)**.** *Let $EPC = (E, F, C, l, A)$ be a relaxed syntactically correct EPC, $N = E \cup F \cup C$ its set of nodes, and M_{EPC} its marking space. If an EPC is sound according to Definition 4.1, all arcs are reachable from some initial marking $i \in I$.*

It is not possible to construct a *livelock* for an EPC. Since we consider relaxed syntactically correct EPCs, each loop must have a split-connector as an exit. If there is a loop in the EPC that has an AND-split as an exit (similar to a token machine in Petri net terms), then a token t_1 can either reach an end arc from the AND-split, or it must deadlock before. In either cases, the token t_2 that is produced in the second iteration of the loop can only be propagated to the input arcs of the node that has t_1 on one of its output arcs. Since the number of arcs between the AND-split and the end arc is finite, the loop will eventually be deadlocked when a token t_i cannot be propagated further from its output arc outside the loop. The consequence of this fact is twofold. Firstly, due to the relaxed syntactical correctness of the EPC and its free-choice behavior, all start arcs which can produce a marking that includes the loop with an AND-split exit must run into a deadlock and are, therefore, not sound. Secondly, due to the safeness of the EPC we only have to look for deadlocks in the reachability graph for verifying soundness. Looking for livelocks is not required. We also summarize this property in the following observation without a proof.

Observation 4.2 (Loops with AND-split exit are not sound)**.** *If there is a loop with an AND-split exit in a relaxed syntactically correct EPC, it is not sound no matter which set of initial markings is considered.*

So-called *contact situations* refer to a marking where there is a token on at least one of the input arcs (token t_2), and another one (token t_1) on at least one of the output arcs of a node n. Due to the safeness property of the EPC (see Section 3.4.4), n cannot fire (cf. e.g. [Kin06]). Now, because of the free-choice property of EPCs, token t_2 on the input arc has the option to follow the other token t_1 in all its firings. Then, t_2 will either be in a deadlock, when t_1 is on an end arc and t_2 on the input arc of the node which is blocked by t_1 on its output arc. Or, there is a join that does not receive the required tokens on other input arcs in order to propagate t_2 to its output. Again, this is a deadlock. Accordingly,

a contact situation implies the option to deadlock. The following observation formulates this fact the other way round.

Observation 4.3 (Contact situation is not sound)**.** *If an $EPC = (E, F, C, l, A)$ is sound, there is no marking reachable that is a contact situation.*

Given the soundness definition, the example EPCs of Figures 3.6 and Figure 3.7 are not sound since the OR-joins block each other. Both EPCs of Figure 3.8 are sound. Finally, both EPCs of Figure 3.9 are not sound because if the token at $a7$ or $a7f$, respectively, exits the loop, the OR-join $c1$ is blocked. In the subsequent section, we show how soundness can be verified based on the reachability graph of an EPC.

4.2 Reachability Graph Verification of Soundness

In this section, we present an approach to verifying soundness based on the reachability graph of an EPC. Since the reachability graph of an EPC is finite and there are no livelocks in an EPCs, we have to consider deadlocks as the starting point of the analysis. In the reachability graph, deadlocks are leaf vertices that are not a final marking.

Definition 4.2 (Deadlock of an EPC)**.** Let $m \in M$ be a marking of an EPC. The marking m is called a deadlock if:

(i) There is no node that can fire in m. Formally:
$\nexists m' \in M : m \rightarrow m'$

(ii) m is not a final marking. Formally:
$m \notin O$

The verification of soundness requires the reachability graph as input. Algorithm 2 shows an object-oriented pseudo code doing the calculation. In particular, we assume that RG is an instance of the class $ReachabilityGraph$ that provides the methods $getLeaves()$ and $getRoots()$. Then, we define $BadLeaves$, $GoodLeaves$, and $GoodRoots$ as instances of a class $MarkingList$ that offers the methods $add(marking)$,

$remove(marking)$, and $missing()$. The first two methods change the markings that are included in the list, the latter method returns a list of nodes that do not have a positive token in any marking of the $MarkingList$. This method basically initializes a list with all start and end arcs and iterates over the markings of the list. For each marking, those arcs with a positive token are deleted. After the iteration, the list includes only the missing arcs. Using this method, we can determine whether I and O cover all start and end arcs. Furthermore, $current, pre,$ and $post$ are instances of class $Marking$ that provides the methods $isLeaf()$, $getPredecessors()$, and $getSuccessors()$. Finally, $predecessorStack$ and $successorStack$ are instances of $Stack$ that provides a $pop()$ and a $push(element)$ method, and $GoodRootSuccessors$ and $BadLeavesPredecessors$ are sets.

The algorithm covers three different phases. In the first phase, the deadlocks are determined and stored in the $BadLeaves$ list of markings (lines 2-6). If there are no deadlocks ($BadLeaves$ is empty), the EPC is sound and the algorithm returns all roots and all leaves of the reachability graph, and two empty sets indicate that there are no start arcs and no end arcs missing in the set of initial markings and final markings. In the second phase (lines 10-23), all predecessor markings of deadlocks are determined. If there is an initial marking found via the $isRoot()$ method, this marking is removed from the $GoodRoots$ list. If the marking is not a root, all those predecessors are added to the stack which have not yet been visited as indicated by the $BadLeavesPredecessors$ list. As a result of this phase, $GoodRoots$ now includes only those initial markings that never result in a deadlock. If $GoodRoots$ was empty, we could stop after line 23 and return two empty lists for good root and leaf elements plus two lists of all start and end arcs. Due to the limited size of the page, we omitted a respective if statement. In the third phase (lines 24-37), we determine those leaves of the reachability graph that can be reached from good roots. The calculation is performed in analogy to the second phase. Finally, line 38 returns the list of good roots and of good leaves as well as a list of start arcs that are not covered by initial markings and end arcs that are not included in final markings. If both arc lists are empty, the EPC is sound.

We implemented Algorithm 2 as an extension to the EPC to Transition System plug-in for ProM (see Section 3.4.6). Figure 4.3 shows the refined EPC of Figure 3.9 on page

Algorithm 2 Pseudo code for verification of soundness

Require: RG

1: $BadLeaves \leftarrow \emptyset, GoodLeaves \leftarrow \emptyset, GoodRoots \leftarrow RG.getRoots()$
2: **for all** $leaf \in RG.getLeaves()$ **do**
3: **if** $\neg leaf.isFinalMarking()$ **then**
4: $BadLeaves.add(leaf)$
5: **end if**
6: **end for**
7: **if** $|BadLeaves| = 0$ **then**
8: **return** $RG.getLeaves(), RG.getRoots(), \emptyset, \emptyset$
9: **end if**
10: $BadLeavesPredecessors \leftarrow \emptyset, predecessorStack \leftarrow BadLeaves$
11: **while** $predecessorStack \neq \emptyset$ **do**
12: $current \leftarrow predecessorStack.pop()$
13: **if** $current.isRoot$ **then**
14: $GoodRoots.remove(current)$
15: **else**
16: **for all** $pre \in current.getPrecessors()$ **do**
17: **if** $pre \notin BadLeavesPredecessors$ **then**
18: $predecessorStack.push(pre)$
19: **end if**
20: **end for**
21: $BadLeavesPredecessors.add(current)$
22: **end if**
23: **end while**
24: $GoodRootSuccessors \leftarrow \emptyset, successorStack \leftarrow GoodRoots$
25: **while** $successorStack \neq \emptyset$ **do**
26: $current \leftarrow successorStack.pop()$
27: **if** $current.isLeaf()$ **then**
28: $GoodLeaves.add(current)$
29: **else**
30: **for all** $post \in current.getSuccessors()$ **do**
31: **if** $post \notin GoodRootSuccessors$ **then**
32: $successorStack.push(post)$
33: **end if**
34: **end for**
35: $GoodRootSuccessors.add(current)$
36: **end if**
37: **end while**
38: **return** $GoodLeaves, GoodRoots, GoodLeaves.missing(), GoodRoots.missing()$

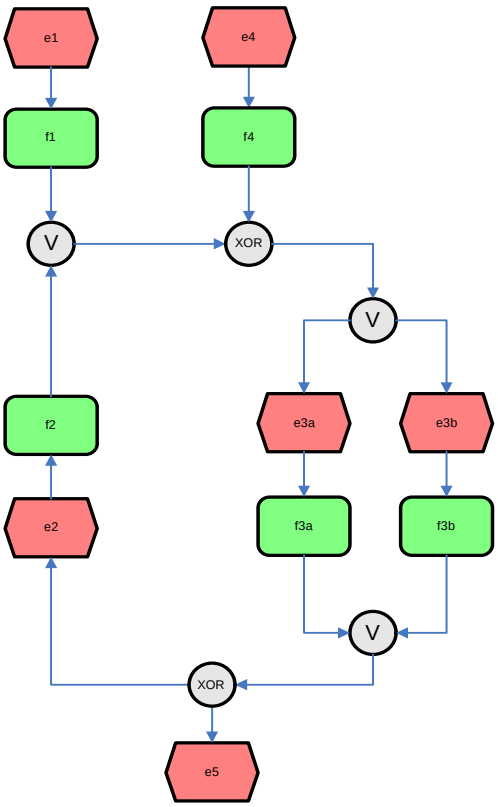


Figure 4.3: The second refinement example EPC in Visio

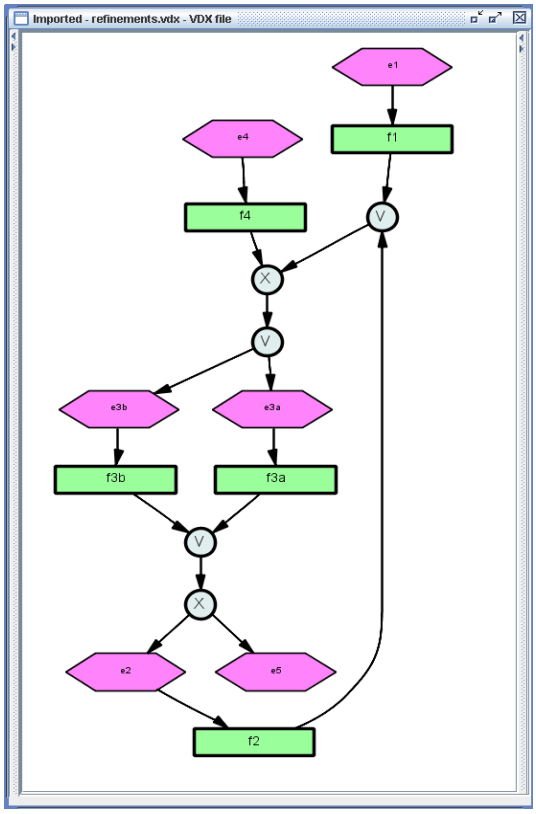


Figure 4.4: The second refinement example EPC loaded in ProM

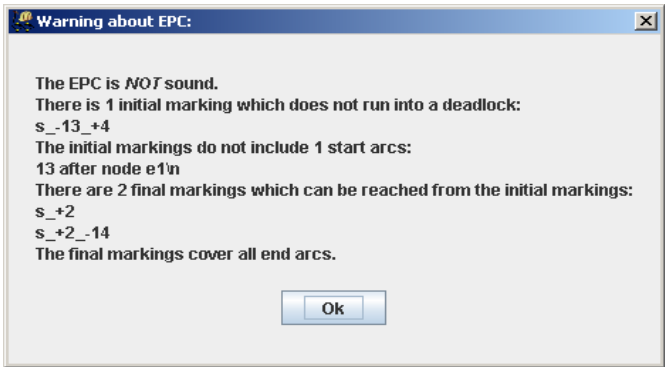


Figure 4.5: Feedback about EPC soundness

58 as it is modelled in Visio. Loading this model in ProM results in a new layout that is displayed in Figure 4.4. If we now use the conversion plug-in to create a transition system, we get a pop-up window that reports the result of the soundness check.[4] For the refinement EPC, there is only one initial marking (positive token after $e4$ and negative token after $e1$) that does not result in a deadlock. Accordingly, the start arc after $e1$ does not have a positive token in any initial marking. Therefore, the EPC is not sound.

Figure 4.6 shows the transition system that is generated for the EPC. Two of the three initial markings are painted with a red border to indicate that they may run into a deadlock. One initial marking has a green border to highlight that it never runs into a deadlock. While the initial marking on the top right of the graph immediately produces a deadlock, the other red initial marking can complete properly, but may also deadlock.

The verification of soundness based on the reachability graph and its implementation in ProM is a powerful tool to identify behavioral problems of EPCs. Still, the potential number of markings grows exponentially with the number of arcs. The reachability graph of a model with 17 arcs, like the refinement example depicted in Figure 4.3, can have up to $|V| = 3^{|A|} = 3^{17} = 129,140,163$ markings as vertices and $|V| \times (|V| - 1) = 16,677,181,570,526,406$ transitions in the worst case. This problem is called the state explosion problem in the Petri nets community (see e.g. [Val98]). One approach to cope with this problem is to apply reduction rules that preserve the property under consideration (see e.g. [Mur89]), i.e. EPC soundness in this context. In the following section, we will investigate in how far reduction rules can be applied for EPCs.

4.3 Verification by Reduction Rules

In the previous section, we presented a verification approach for EPC soundness based on the reachability graph. Similar to Petri nets, the concurrency of functions and events can lead to a performance problem due to the state explosion. In this section, we focus on an approach based on reduction rules to increase the performance of the verification process. First, we revisit related work on reduction rules for business process models

[4] The result is also written to the message panel at the bottom of ProM.

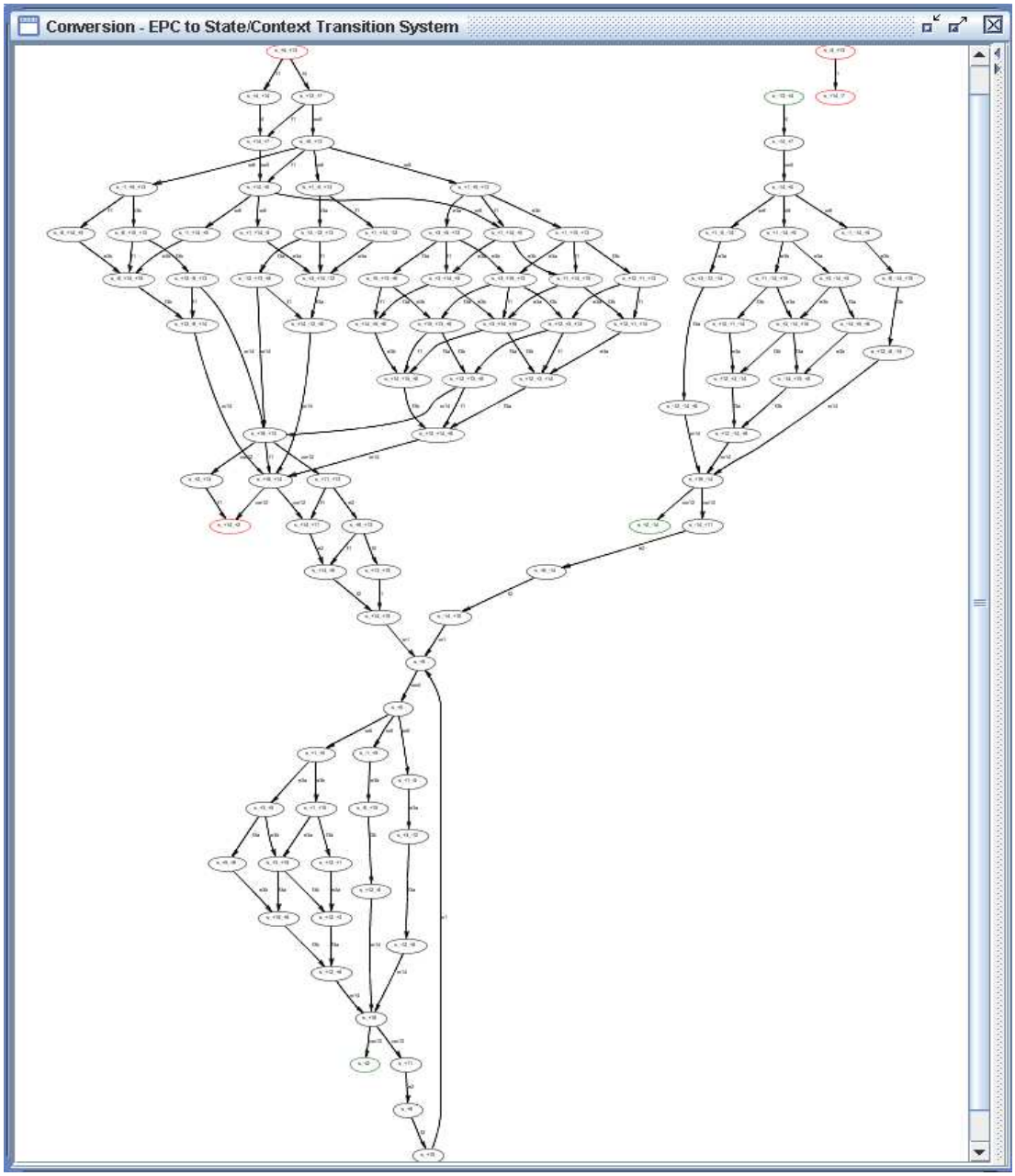


Figure 4.6: Transition System of the second refinement example EPC

(Section 4.3.1). After that, we present a set of reduction rules for EPCs that, on one hand, extends existing rule sets and that is, on the other hand, easy to apply for a given model (Section 4.3.2). In the following pages, we use the term *reduction kit* to refer to a set of reduction rules (cf. [Esp94]). It must be mentioned that our EPC reduction kit is sound but not complete, i.e., the fact that an EPC is not completely reduced does not provide an answer to the verification question, i.e., there may be errors, but this is not sure. Still, even if the model is not completely reduced, the reachability graph verification is more efficient than for the unreduced EPC. Furthermore, if the EPC is reduced to the trivial model and no errors are recorded, it is sound. In Section 4.3.3 we present a reduction algorithm and a respective implementation in the *xoEPC* program. After that, Section 4.3.4 illustrates the application of *xoEPC* in the analysis of the SAP reference model. In particular, we present which rules are used how often, how many EPCs could be reduced, and how many errors were found.

4.3.1 Related Work on Reduction Rules

The state explosion problem is one of the motivations for considering reduction rules for Petri nets. A set of six reduction rules that preserve liveness, safeness, and boundedness of a Petri net is introduced in *Berthelot* [Ber86, Ber87] and summarized in *Murata* [Mur89, p.553]. Yet, this set of rules is not complete, i.e., there are live, safe, and bounded Petri nets that cannot be reduced to the trivial model by these rules. Figure 4.7 shows an illustration of the six reduction rules including (a) *fusion of series places*, (b) *fusion of series transitions*, (c) *fusion of parallel places*, (d) *fusion of parallel transitions*, (e) *elimination of self-loop places*, and (f) *elimination of self-loop transitions*. For the Petri net class of free choice nets, *Esparza* shows that there exists a complete reduction kit including rules for fusion of places and transitions similar to (a) and (b) of *Murata* and two linear dependency rules to eliminate nonnegative linearly dependent places and transitions [Esp94, DE95]. By showing that soundness corresponds to liveness and boundedness of the short-circuited net, *Van der Aalst* makes the reduction kit of *Murata* applicable to the analysis of workflow nets [Aal97].

In a different stream of research, *Sadiq & Orlowska* discuss the applicability of re-

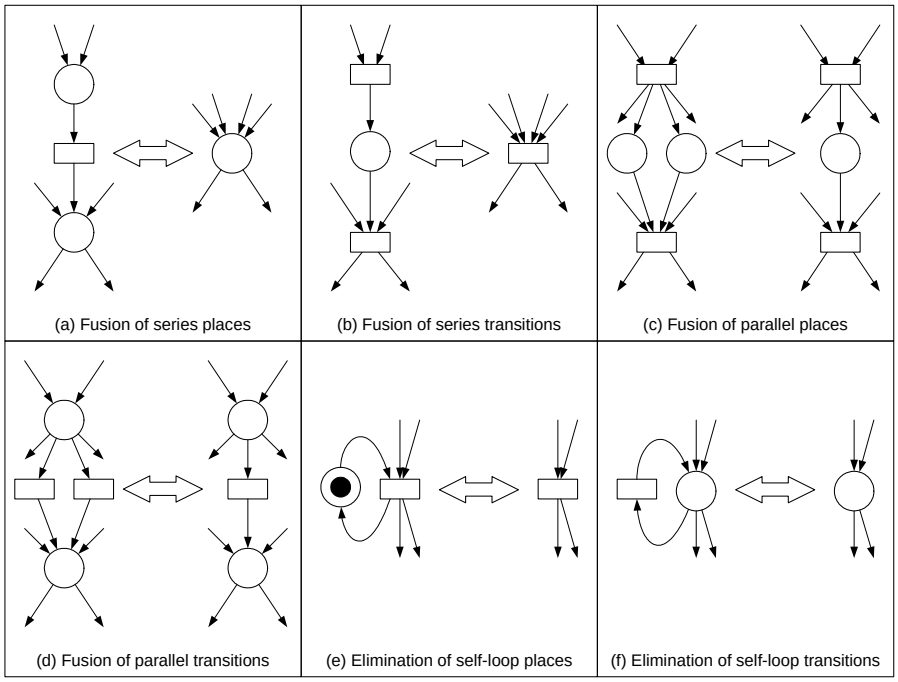


Figure 4.7: Six reduction rules to preserve liveness, safeness, and boundedness [Mur89]

duction rules for business process models that are defined in a language called workflow graphs [SO96, SO99, SO00]. They provide a kit including (a) the *adjacent reduction rule* to merge parallel splits and joins, (b) the *closed reduction rule* to eliminate redundant synchronization arcs, and (c) the *overlapped reduction rule* that eliminates a proper block with one XOR-split, multiple AND-splits, multiple XOR-joins, and one AND-join. *Lin et al.* show that the reduction kit of *Sadiq & Orlowska* is not complete by giving a counter example [LZLC02]. They propose a new reduction kit including seven rules: (a) the *terminal reduction rule* to eliminate sequential start and end nodes, (b) the *sequential reduction rule* to eliminate sequences, (c) the *adjacent reduction rule* of *Sadiq & Orlowska*, (d) the *closed reduction rule* of *Sadiq & Orlowska*, (e) the *choice-convergence reduction rule* to move a choice out of a parallel structure, (f) the *sychronizer-convergence reduction rule* that moves a sychronization out of a choice structure, and (g) the *merge-fork reduction rule*, which actually replaces a simple structure with a complicated one. *Van der Aalst, Hirnschall, and Verbeek* showed that the original reduction kit is not complete and question approaches to continue adding rules for counter examples [AHV02]. They

use a completely different approach building on well-known Petri net results. By providing a mapping to Petri nets, they show that the resulting net is free choice [AHV02]. On the one hand, this makes the complete reduction kit of *Esparza* applicable. On the other hand, basic Petri net analysis techniques and tools can be applied and soundness can be checked in polynomial time.

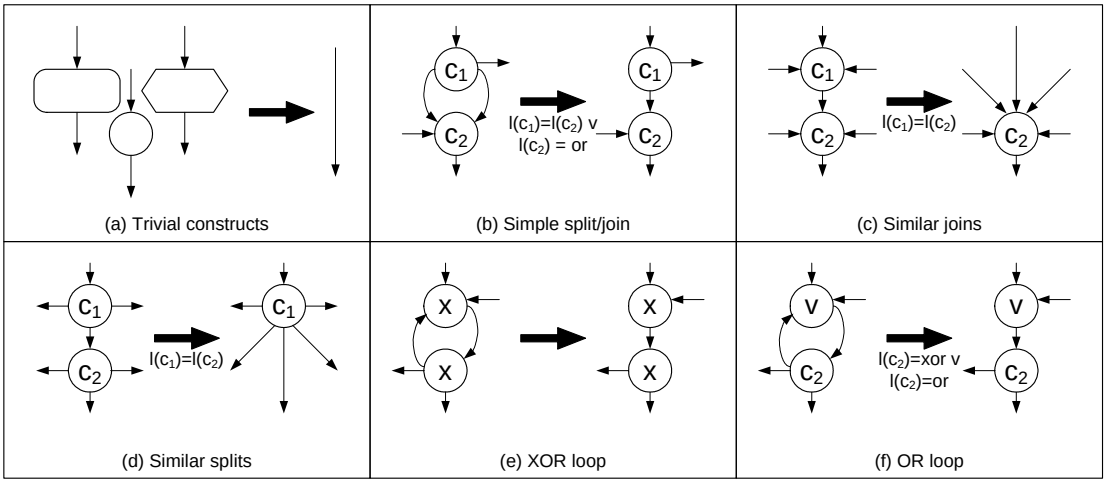


Figure 4.8: Six reduction rules of [DAV05]

A set of reduction rules for flat EPCs was first mentioned in *Van Dongen, Van der Aalst, and Verbeek* in [DAV05]. The idea is to eliminate those structures of an EPC that are trivially correct for any semantics formalization. Rule (a) deletes sequential elements from the EPC, i.e. elements with one input and one output arc. Rule (b) merges multiple parallel arcs between connectors of the same type. This might result in connectors with one-one cardinality so that rule (a) can be applied. Rules (c) and (d) merge consecutive join and split connectors of the same type. Rule (e) eliminates the backward arc of a simple XOR-loop. Finally, rule (f) reduces OR-loops based on the assumption that only arcs from outside the loop are synchronized (cf. [Don07]). The authors use this reduction kit to derive a more compact EPC for further analysis: the domain expert then has to specify the set of allowed initial markings for the reduced EPC. This information is used in a coverability analysis of a Petri net representation of the EPC to identify structural problems. The approach has been implemented as an analysis plug-in which is shipped with the standard distribution of ProM.

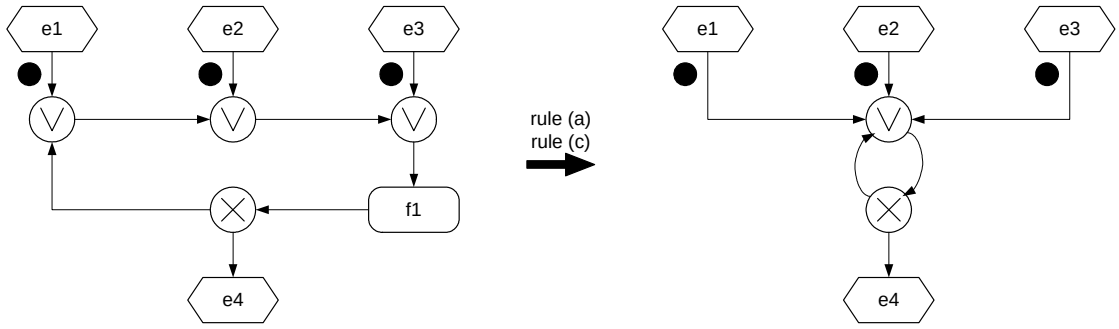


Figure 4.9: Unclean and deadlocking EPC

While the rules by *Van Dongen et al.* are indeed helpful to reduce the complexity of the verification problem, it is possible to reduce erroneous models (assuming the semantics of this thesis) with the combination of rule (c) and (f). Figure 4.9 shows an EPC that is unclean under the semantics of *Kindler* [Kin06], and in deadlock under the semantics of *Wynn et al.* [WAHE06] as well as under the semantics presented in Section 3.4.4. Still, this EPC can be easily reduced by first merging the three OR-joins with rule (c) and then eliminating the OR-loop with rule (f). The further application of rule (a) yields a trivial EPC consisting only of three start events, one OR-join, and one end event. Since loops without XOR-entries deadlock following our novel semantics, we have to consider structured loops with OR-entries as error cases.

In [WVA+06a] *Wynn, Verbeek, Van der Aalst, Ter Hofstede, and Edmond* discuss reduction rules for Reset nets, a Petri net class that offers so-called reset arcs which clean tokens from the net. Their reduction kit of seven reduction rules is mainly inspired by rules for Petri nets by *Murata* [Mur89] and for free-choice nets by *Esparza* [Esp94, DE95]. The rules 1 to 4, i.e. *fusion of series places* and of *series transitions* as well as *fusion of parallel places* and *parallel transitions*, can be directly related to the rules of *Murata* and *Esparza*. Rule 5 (*abstraction rule*) removes a sequence of a place s and a transition t where s is the only input of t and t the only output of s, and there is no direct connection between the inputs of s and the outputs of t. Rule 6 for *self-loop transitions* matches the self-loop rule of *Murata*. Finally, rule 7 (*fusion of equivalent subnets*) allows identical parts of the net to be merged.

Based on a Reset net formalization *Wynn, Verbeek, Van der Aalst, Ter Hofstede, and*

Edmond define a reduction kit for YAWL [WVA+06b]. Furthermore, they prove that it preserves soundness by constructing respective Reset net reductions. Several rules are defined for YAWL nets including *fusion of series, parallel, and alternative conditions*; *fusion of series, parallel, and alternative tasks*; *elimination of self-loop tasks*, *fusion of AND-split and AND-join tasks*; *fusion of XOR-split and XOR-join tasks*; *fusion of an OR-join and another task*; and *fusion of incoming edges to an OR-join*. The last two rules differ from the other rules since they are not explicitly proved based on Reset net rules. This is because the enabling rule of an OR-join depends on the reachability analysis of the YAWL net (see [WAHE06]).

4.3.2 A Reduction Kit for EPCs

In this section, we take the work of *Van Dongen et al.* as a starting point and introduce novel reduction rules. Since we want to increase the performance of verification, we are interested in rules which involve as few directly connected nodes as possible. Furthermore, we are not interested in the completeness of the rules. For each reduction rule, we have to show that it does not introduce a deadlock into the EPC that affects property (ii) of EPC soundness. For rules that reduce the set of initial or final markings, we have to show that properties (i) to (iii) of EPC soundness are not violated. Additionally, both the source and the target EPC have to fulfill the requirements of relaxed syntactically correctness.

In this context, a *reduction rule T* is a binary relation that transforms a source EPC_1 to a simpler target EPC_2 that has less nodes and/or arcs (cf. e.g. [Esp94]). Furthermore, we associate an index function f_A with an EPC for keeping track of multiple arcs that might be derived in the reduction process. A reduction rule is bound to a *condition* that defines for which arcs and nodes it is applicable. Furthermore, we define the *construction* of the target EPC and *error cases* for several of the rules. Our strategy is to record errors, apply the reduction, and continue with the reduced model to potentially find further errors. Figure 4.10 gives an overview of the reduction rules that we discuss on the following pages.

Some of the reduction rules we will define might introduce already existing arcs to

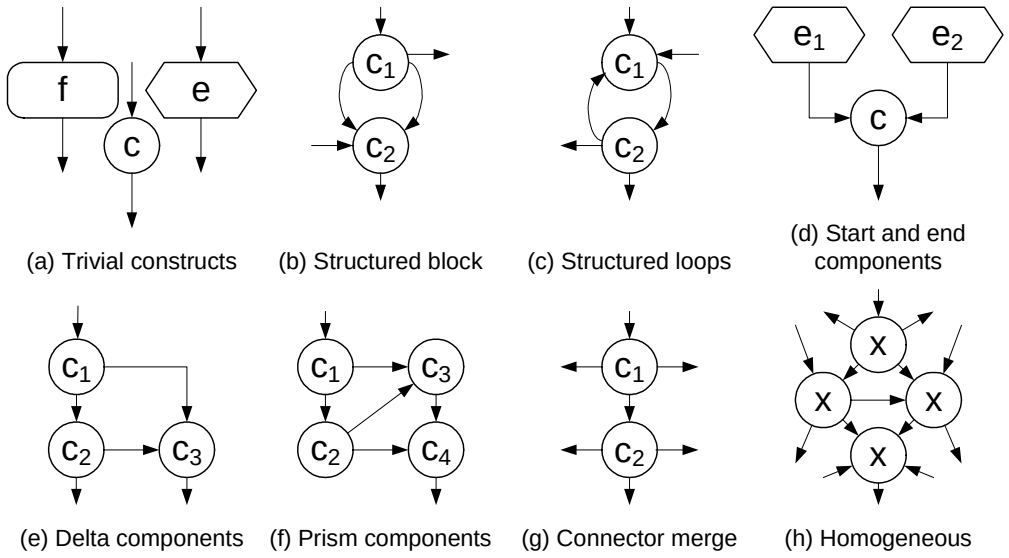


Figure 4.10: Overview of patterns that are addressed by EPC reduction rules

the reduced EPC. Consider the EPC on the left hand side of Figure 4.11. The reduction of trivial constructs that we will introduce afterwards first replaces the function $f1$ and its input and output arcs (2 and 4) by a single arc (7). Then, the same procedure is applied for function $f2$. The problem in this case is that an arc between the AND-split and the XOR-join already exists. We use the index set I_A, the index function f_A and the count function ξ to keep track of added arcs that already exist. While the EPC on the right hand side has only one arc between the two connectors, there are two indices 7 and 8 in the index set I_A pointing at it. Without indices, we lose the information that the two connectors are problematic. The indexing mechanism allows us to define several rules in a more simple way as opposed to parameterizing each rule to deal with potentially multiple arcs. Note that the EPC semantics can be easily adapted to deal with the index set extension.

Definition 4.3 (Index Set and Index Function for Arcs)**.** Let $EPC = (E, F, C, l, A)$ be a relaxed syntactically correct EPC. Then $I_A \subset \mathbb{N}$ is an index set such that $f_A : I_A \rightarrow A$ is a totally surjective function mapping all elements of I_A onto the set of arcs. Furthermore, we define the function $\xi : I_A \times A \rightarrow \mathbb{N}$ such that for $a \in A, I_A : \xi(a, I_A) = |\{x \in I_A \mid f_A(x) = a\}|$.

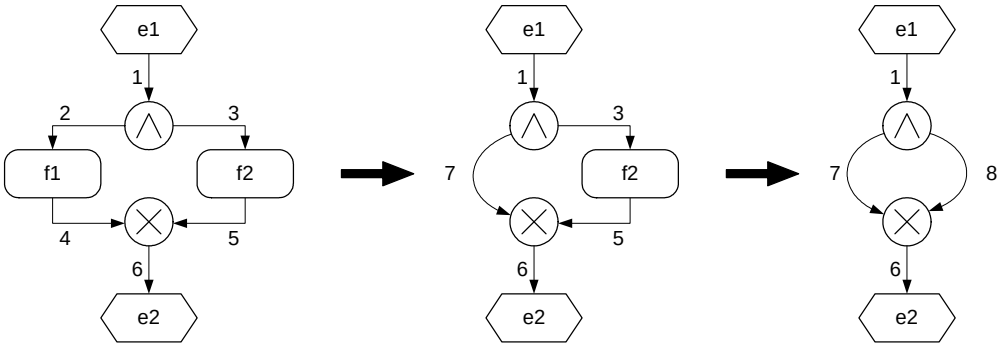


Figure 4.11: Reduction producing arcs that already exist

On the following pages we define the different reduction rules. These definitions build on reduction relations T_x with $x \in \{a, b, c, d, d1a, d1b, d2a, d2b, e, f, g, h\}$ referring to the different rule types as depicted in Figure 4.10.

Trivial Constructs

The reduction of trivial constructs allows for eliminating sequential nodes from EPC which have one input and one output arc. These nodes do not cause deadlock problems and neither does their removal. The rule is similar to the fusion of series places and transitions for Petri nets by *Murata* [Mur89] and previously defined by *Van Dongen et al.* [DAV05]. Figure 4.12 illustrates the rule.

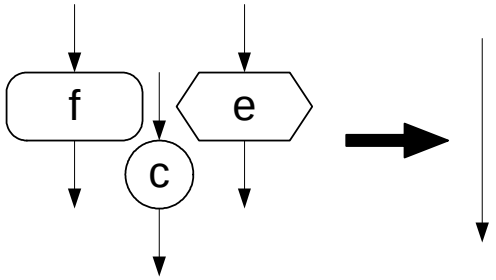


Figure 4.12: Reduction of trivial constructs

Definition 4.4 (Reduction of Trivial Constructs)**.** Let EPC_1 and EPC_2 be two relaxed syntactically correct EPCs with the respective index sets I_{A_1} and I_{A_2}, index and count

functions f_{A_1}, f_{A_2}, ξ_1, and ξ_2. The pair $((EPC_1, I_{A_1}), (EPC_2, I_{A_2})) \in T_a$ if the following conditions hold for a node $n \in N_1$ of EPC_1, and if EPC_2 can be constructed from EPC_1 as follows:

Condition:

1) there exists $n \in N_1$ such that
   $|n_{in}| = |n_{out}| = 1 \wedge$
   $n_{in} = \{(v, n)\} \wedge \xi_1((v, n), I_{A_1}) = 1 \wedge$
   $n_{out} = \{(n, w)\} \wedge \xi_1((n, w), I_{A_1}) = 1.$

Construction:

2) $E_2 = E_1 \setminus \{n\}$
3) $F_2 = F_1 \setminus \{n\}$
4) $C_2 = C_1 \setminus \{n\}$
5) $A_2 = (A_1 \cup \{(v, w)\}) \setminus \{(v, n), (n, w)\}$
6) Introduce i_{vw} such that $i_{vw} \in \mathbb{N} \setminus I_{A_1} \wedge$
   $I_{A_2} = (I_{A_1} \cup \{i_{vw}\}) \setminus \{i \in I_{A_1} \mid f_{A_1}(i) = (v, n) \vee f_{A_1}(i) = (n, w)\} \wedge$
   $\forall i \in I_{A_2} \setminus \{i_{vw}\} : f_{A_2}(i) = f_{A_1}(i) \wedge$
   $f_{A_2}(i_{vw}) = (v, w)$

There are no error cases.

It can be shown that the rule preserves relaxed syntactical correctness. If all nodes of EPC_1 were on a path from a start to an end node, this must obviously still hold for EPC_2 after reduction. Furthermore, the cardinality restrictions still hold since there is no node beyond the deleted node that has a different cardinality after reduction. Finally, the only case where the rule can produce self-arcs is if an undesirable structure exists such that not every node is on a path from a start to an end node (cf. Figure 3.3 on page 41). Yet, according to relaxed syntactical correctness, such a structure is not allowed .

Structured Blocks

Structured blocks include a split and a join connector with multiple arcs from the split to the join (see Figure 4.13). To be concise, the multiple arcs are an illustration of the fact that there are multiple indices in I_{A_1} pointing at the arc from c_1 to c_2. Such structured

blocks usually appear when parallel or alternative sequences are reduced by other rules. If the type of both connectors is equivalent, the rule matches the parallel place and transition reduction rules of *Murata* [Mur89] and the structured component rule of *Van Dongen et al.* [DAV05]. In these cases, it is safe to fuse the parallel arcs, i.e. the multiple indices are replaced by a single index. Still, there are four problematic cases: if c_1 is an XOR or an OR and c_2 is an AND, the process can run into a deadlock which implies that it is not sound. Furthermore, if c_2 is an XOR and c_1 is an AND or an OR, there is a lack of synchronization which can result in contact situations. Again, the process is not sound. In these cases, we record the error and continue searching for further errors in the reduced model.

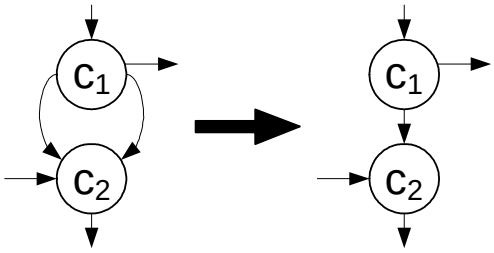


Figure 4.13: Reduction of structured blocks

Definition 4.5 (Reduction of Structured Block)**.** Let EPC_1 and EPC_2 be two relaxed syntactically correct EPCs with the respective index sets I_{A_1} and I_{A_2}, index and count functions f_{A_1}, f_{A_2}, ξ_1, and ξ_2. The pair $((EPC_1, I_{A_1}), (EPC_2, I_{A_2})) \in T_b$ if the following conditions hold for a pair of connectors $c_1, c_2 \in C_1$ of EPC_1, and if EPC_2 can be constructed from EPC_1 as follows:

Condition:

1) there exists $c_1, c_2 \in C_1$ such that
   $c_1 \neq c_2 \ \wedge \ (c_1, c_2) \in A_1 \ \wedge \ \xi((c_1, c_2), I_{A_1}) > 1$

Construction:

2) $E_2 = E_1$
3) $F_2 = F_1$
4) $C_2 = C_1$
5) $A_2 = A_1$
6) Introduce i_{cc} such that $i_{cc} \in \mathbb{N} \setminus I_{A_1} \wedge$

$I_{A_2} = I_{A_1} \cup \{i_{cc}\} \setminus \{i \in I_{A_1} \mid f_{A_1}(i) = (c_1, c_2)\} \wedge$
$\forall i \in I_{A_2} \setminus \{i_{cc}\} : f_{A_2}(i) = f_{A_1}(i) \wedge$
$f_{A_2}(i_{cc}) = (c_1, c_2)$

Error Cases:

7) $l(c_1) = xor \wedge l(c_2) = and$ (Deadlock)
8) $l(c_1) = or \wedge l(c_2) = and$ (Potential Deadlock)
9) $l(c_1) = and \wedge l(c_2) = xor$ (Lack of Synchronization)
10) $l(c_1) = or \wedge l(c_2) = xor$ (Potential Lack of Synchronization)

Obviously, the rule preserves the cardinality restrictions of all nodes and the coherence restriction of relaxed syntactical correctness.

Structured Loops

Structured loops include a join as an entry to the loop and a split as exit, with one arc from the join to the split, and one in the opposite direction (see Figure 4.14). If the type of both connectors is XOR, the rule is similar to the loop elimination rules by *Murata* [Mur89] and the XOR-loop rule of *Van Dongen et al.* [DAV05]. In these cases, it is safe to delete the back arc. Still, there are problematic cases: if c_1 is not an XOR, the loop cannot be entered because the entry-join deadlocks. Furthermore, if c_2 is not an XOR, the process can run into a contact situation (cf. Observation 4.2). In both cases, the process is not sound and an error is recorded before applying the reduction rule.

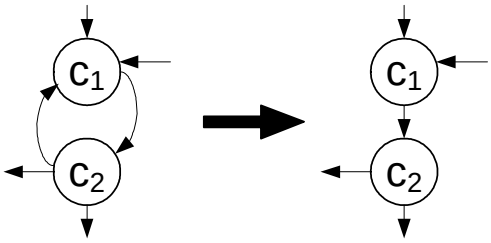


Figure 4.14: Reduction of structured loops

Definition 4.6 (Reduction of Structured Loop)**.** Let EPC_1 and EPC_2 be two relaxed syntactically correct EPCs with the respective index sets I_{A_1} and I_{A_2}, index and count

functions f_{A_1}, f_{A_2}, ξ_1, and ξ_2. The pair $((EPC_1, I_{A_1}), (EPC_2, I_{A_2})) \in T_c$ if the following conditions hold for a pair of connectors $c_1, c_2 \in C_1$ of EPC_1, and if EPC_2 can be constructed from EPC_1 as follows:

Condition:

1) there exists $c_1, c_2 \in C_1$ such that
   $c_1 \neq c_2 \ \wedge \ (c_1, c_2) \in A_1 \wedge (c_2, c_1) \in A_1 \wedge$
   $\xi((c_1, c_2), I_{A_1}) = 1 \ \wedge \ \xi((c_2, c_1), I_{A_1}) = 1$

Construction:

2) $E_2 = E_1$
3) $F_2 = F_1$
4) $C_2 = C_1$
5) $A_2 = A_1 \setminus \{(c_2, c_1)\}$
6) $I_{A_2} = I_{A_1} \setminus \{i \in I_{A_1} \mid f_{A_1}(i) = (c_2, c_1)\} \wedge$
   $\forall i \in I_{A_2} : f_{A_2}(i) = f_{A_1}(i)$

Error Cases:

7) $l(c_1) \neq xor$ (Loop cannot be entered)
8) $l(c_2) \neq xor$ (Potential contact situation)

Again, the rule preserves the cardinality restrictions of all nodes and the coherence restriction of relaxed syntactical correctness.

Start and End Components

A specific verification problem of EPCs, in contrast to workflow nets, is that they may have multiple start and end nodes. Models from practice sometimes have more than 20 start events (i.e. $|I_{EPC}| = 2^{20} - 1$ initial markings). In this case, the reduction of these nodes becomes a critical issue to make verification feasible regarding the complexity (cf. [VA06, MMN+06c]). In this section, we introduce reduction rules for so-called structured and unstructured start and end components. Both these rule sets aim to reduce the number of start and end events without affecting the soundness of the EPC. For unstructured start and end components we distinguish a set for connectors not on a cycle and connectors on a cycle.

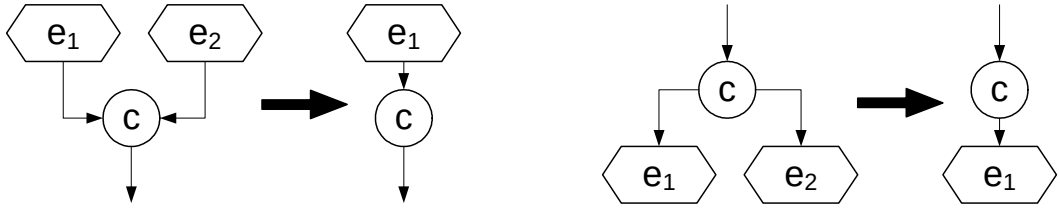


Figure 4.15: Reduction of structured start and end components

Figure 4.15 illustrates the reduction rules for *structured start and end components*. A structured start component contains two start events and one join connector, while a structured end component has one split connector and two end events. In both cases, the second event and the respective arc can be eliminated without affecting the overall soundness. Furthermore, there are no error cases.

Definition 4.7 (Reduction of Structured Start and End Components)**.** Let EPC_1 and EPC_2 be two relaxed syntactically correct EPCs with the respective index sets I_{A_1} and I_{A_2}, index and count functions f_{A_1}, f_{A_2}, ξ_1, and ξ_2. The pair $((EPC_1, I_{A_1}), (EPC_2, I_{A_2})) \in T_d$ if the following conditions hold for two start or two end events $e_1, e_2 \in E_1$ and a connector $c \in C_1$ of EPC_1, and if EPC_2 can be constructed from EPC_1 as follows:
Condition:

1) there exists $c \in C_1, e_1, e_2 \in E_1$ and $a_1, a_2 \in A_1$ such that
   $e_1 \neq e_2 \wedge$
   $a_1 = (e_1, c), a_2 = (e_2, c) \in A_1 \vee a_1 = (c, e_1), a_2 = (c, e_2) \in A_1 \wedge$
   $\xi(a_1, I_{A_1}) = 1 \ \wedge \ \xi(a_2, I_{A_1}) = 1$

Construction:

2) $E_2 = E_1 \setminus \{e_2\}$
3) $F_2 = F_1$
4) $C_2 = C_1$
5) $A_2 = A_1 \setminus \{a_2\}$
6) $I_{A_2} = I_{A_1} \setminus \{i \in I_{A_1} \mid f_{A_1}(i) = a_2\} \wedge$
   $\forall i \in I_{A_2} : f_{A_2}(i) = f_{A_1}(i)$

There are no error cases.

The rule preserves the cardinality restrictions of all nodes, and the restriction that all

nodes must be on a path from a start to an end node. Therefore, the reduced model is relaxed syntactical correct.

Figure 4.16 shows an EPC from the SAP Reference Model that illustrates the correctness of the reduction rule. In particular, two observations can be made. Firstly, the two start events on the top left-hand side can be merged in such a way that afterwards the subsequent AND-join is deleted, too. In this case, a token on the start arc in the reduced EPC basically represents the case where there are tokens on each of the two start arcs in the unreduced model. Both cases lead to the same behavior once the subsequent AND-split connector is reached. Secondly, the red arrows highlight three errors of the EPC, implying that the model is not sound. In each of these cases, end events are connected to one XOR-split while, later, there is an AND-join that might need a token to continue processing. The reduction rule merges the two end events into one in each case, for instance, the *Purchase requisition/order to be created* and the *Funds reservation rejected/budget exceeded* end events after the first problematic XOR-split. This way, the erroneous behavior is preserved since it is not important *how many* XOR-jumps out of the process exist, but only *if* there exists one. The identification of such problematic jumps is subject of the next reduction rule for unstructured start and end components.

Figure 4.17 shows the reduction rules for *unstructured start and end components* that are applicable for connectors that are *not on a cycle*. In case (a), there is an AND-split connector c_1 followed by an end event. Since this end event is reachable if the connector is reachable, we can delete e_1 and consider the input arc of c_1 only. Furthermore, the output arc not pointing to e_1 will always be receive control via c_1 no matter whether there is an arc to e_1 or not. If c_1 is not an AND-split but an OR- or an XOR-split, we can consider case (b), where the structure is extended with a join-connector c_2 and a start event e_2. If c_2 is an AND-join it might not get control from the (X)OR-split c_1, which implies that the structure is not sound. In this case, an error is recorded and the branch to e_1 as the reason is deleted. This pattern appears three times in Figure 4.16, as previously discussed.

Definition 4.8 (Reduction of Unstructured Acyclic Start and End Components (a))**.** Let EPC_1 and EPC_2 be two relaxed syntactically correct EPCs with the respective index sets I_{A_1} and I_{A_2}, index and count functions f_{A_1}, f_{A_2}, ξ_1, and ξ_2. The pair

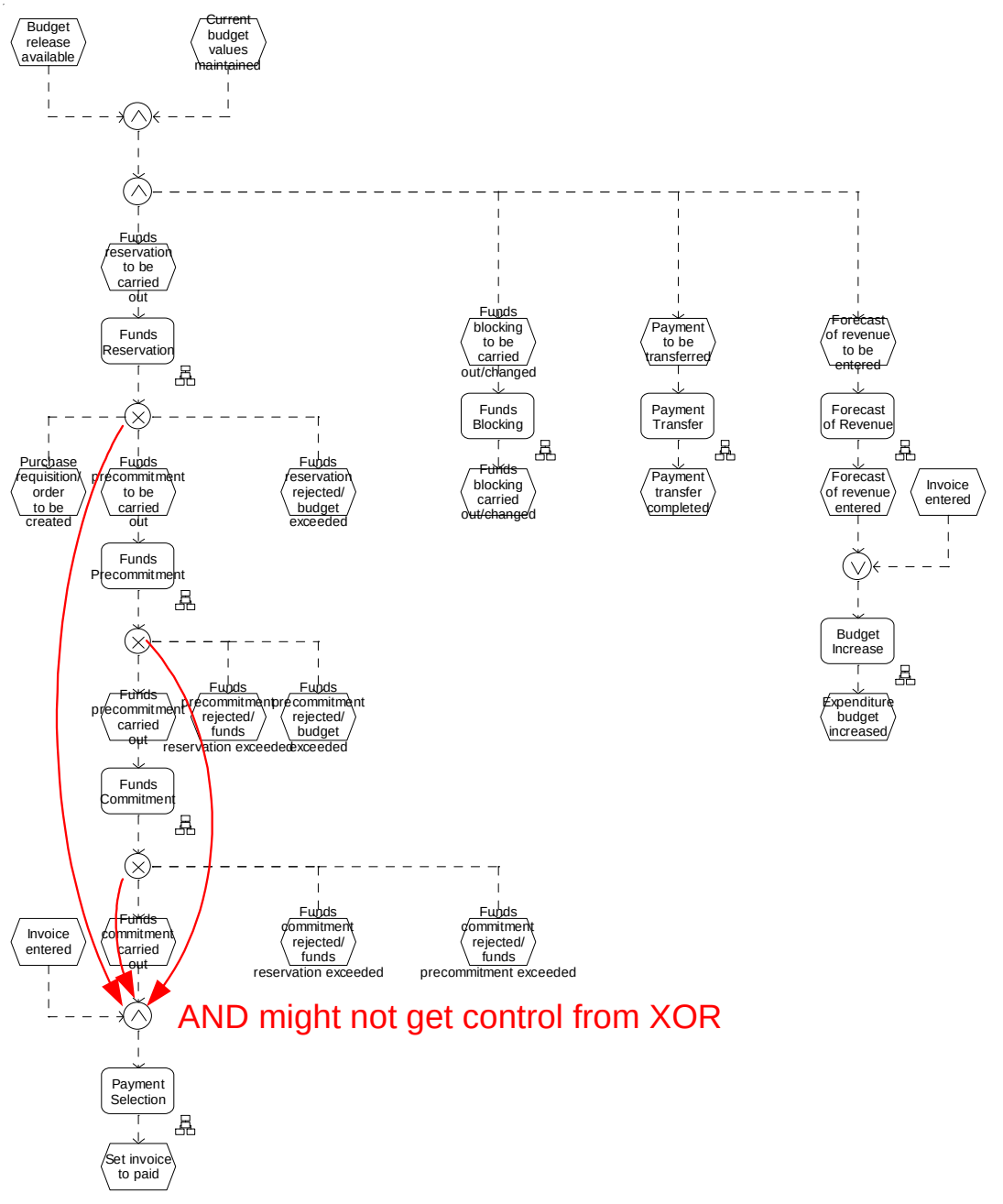


Figure 4.16: Financial Accounting – Funds Management – Budget Execution

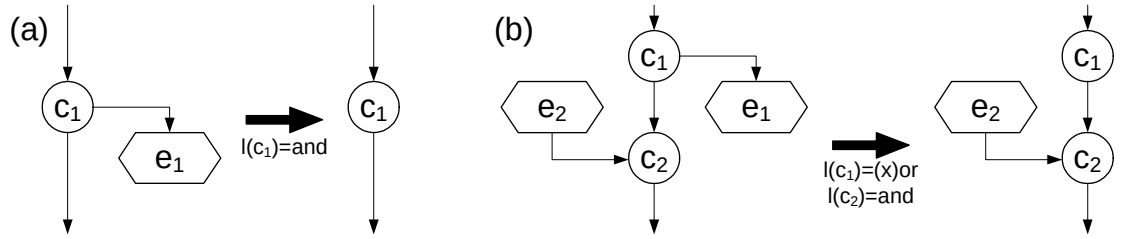


Figure 4.17: Reduction of unstructured start and end components, not on cycle

$((EPC_1, I_{A_1}), (EPC_2, I_{A_2})) \in T_{d1a}$ if the following conditions hold for one end event $e_1 \in E_1$ and a connector $c_1 \in C_1$ of EPC_1, and if EPC_2 can be constructed from EPC_1 as follows:

Condition:

1) there exists $c_1 \in C_1, e_1 \in E_1$ and $a_1 \in A_1$ such that
   $e_1\bullet = \emptyset \wedge$
   $a_1 = (c_1, e_1,) \in A_1 \ \wedge \ l(c_1) = and \wedge$
   $\xi(a_1, I_{A_1}) = 1 \wedge$
   $\nexists n \in N \setminus \{c_1\} : c_1 \hookrightarrow n \hookrightarrow c_1$

Construction:

2) $E_2 = E_1 \setminus \{e_1\}$
3) $F_2 = F_1$
4) $C_2 = C_1$
5) $A_2 = A_1 \setminus \{a_1\}$
6) $I_{A_2} = \{i \in I_{A_1} \mid f_{A_1}(i) \neq a_1\} \wedge$
   $\forall i \in I_{A_2} : f_{A_2}(i) = f_{A_1}(i)$

There are no error cases.

Definition 4.9 (Reduction of Unstructured Acyclic Start and End Components (b))**.** Let EPC_1 and EPC_2 be two relaxed syntactically correct EPCs with the respective index sets I_{A_1} and I_{A_2}, index and count functions f_{A_1}, f_{A_2}, ξ_1, and ξ_2. The pair $((EPC_1, I_{A_1}), (EPC_2, I_{A_2})) \in T_{d1b}$ if the following conditions hold for one start or one end event $e_1, e_2 \in E_1$ and a connector $c_1, c_2 \in C_1$ of EPC_1, and if EPC_2 can be constructed from EPC_1 as follows:

Condition:

1) there exists $c_1, c_2 \in C_1, e_1, e_2 \in E_1$ and $a_1, a_2, a_3 \in A_1$ such that
   $e_1 \neq e_2 \ \wedge \ c_1 \neq c_2 \wedge$
   $e_1\bullet = \emptyset \ \wedge \ \bullet e_2 = \emptyset$
   $l(c_1) \neq and \ \wedge \ l(c_2) = and \wedge$
   $a_1 = (c_1, e_1,), a_2 = (e_2, c_2), a_3 = (c_1, c_2) \in A_1 \ \wedge \ (c_2, c_1) \notin A_1 \wedge$
   $\xi(a_1, I_{A_1}) = 1 \ \wedge \ \xi(a_2, I_{A_1}) = 1 \ \wedge \ \xi(a_3, I_{A_1}) = 1 \wedge$
   $\nexists n \in N \setminus \{c_1, c_2\} : c_1 \hookrightarrow n \hookrightarrow c_1 \ \vee \ c_2 \hookrightarrow n \hookrightarrow c_2$

Construction:

2) $E_2 = E_1 \setminus \{e_1\}$
3) $F_2 = F_1$
4) $C_2 = C_1$
5) $A_2 = A_1 \setminus \{a_1\}$
6) $I_{A_2} = I_{A_1} \setminus \{i \in I_{A_1} \mid f_{A_1}(i) = a_1\} \wedge$
   $\forall i \in I_{A_2} : f_{A_2}(i) = f_{A_1}(i)$

Error Cases:

7) The reduction rule always yields an error since c_2 might not get control from c_1.

Both these rules for unstructured acyclic start and end components preserve the cardinality restrictions of all nodes and the coherence restriction of relaxed syntactical correctness.

Figure 4.18 shows the reduction rules for *unstructured start and end components* that are applicable for connectors that are *on a cycle*. By considering two connectors, we make sure that the reduction does not delete the last exit or the last entry point of a loop. Such a reduction would violate the relaxed syntactical correctness of the reduced EPC. In case (a) there are two exit-connectors $c_1, c_2 \in C_1$ to a cyclic part of the EPC and at least one of them, i.e. $c_{nonxor} \in \{c_1, c_2\}$, is not of type XOR. This implies that if the loop is executed multiple times, then the non-XOR-connector c_{nonxor} will repeatedly create tokens at the same exit of the loop. According to our definition of safe semantics, this leads to contact situations and unsound behavior of the EPC. Therefore, the reduction rule deletes the reason for this error, i.e. the end-event from the non-XOR-connector c_{nonxor}. In case (b), there are two entry connectors to a loop. If there is an AND-join among them (c_{and}), the EPC is not sound since there will be a token missing on the start arc in the

second execution of the loop. Therefore, the start-event leading to the AND-join c_{and} is deleted and an error is recorded.

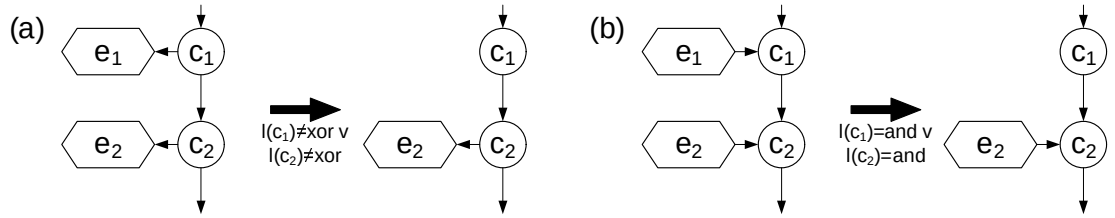


Figure 4.18: Reduction of unstructured start and end components, not on cycle

Definition 4.10 (Reduction of Unstructured Cyclic Start and End Components (a))**.** Let EPC_1 and EPC_2 be two relaxed syntactically correct EPCs with the respective index sets I_{A_1} and I_{A_2}, index and count functions f_{A_1}, f_{A_2}, ξ_1, and ξ_2. The pair $((EPC_1, I_{A_1}), (EPC_2, I_{A_2})) \in T_{d2a}$ if the following conditions hold for one start or one end event $e_1, e_2 \in E_1$ and a connector $c_1, c_2 \in C_1$ of EPC_1, and if EPC_2 can be constructed from EPC_1 as follows:

Condition:

1) there exists $c_1, c_2 \in C_1, e_1, e_2 \in E_1$ and $a_1, a_2, a_3 \in A_1$ such that
   $e_1 \neq e_2 \ \wedge \ c_1 \neq c_2 \wedge$
   $e_1\bullet = \emptyset \ \wedge \ e_2\bullet = \emptyset$
   $a_1 = (c_1, e_1,), a_2 = (c_2, e_2), a_3 = (c_1, c_2) \in A_1 \ \wedge \ (c_2, c_1) \notin A_1 \wedge$
   $\xi(a_1, I_{A_1}) = 1 \ \wedge \ \xi(a_2, I_{A_1}) = 1 \ \wedge \ \xi(a_3, I_{A_1}) = 1 \ \wedge$
   $\exists n_1 \in N \setminus \{c_1\} : c_1 \hookrightarrow n_1 \hookrightarrow c_1 \wedge$
   $\exists n_2 \in N \setminus \{c_2\} : c_2 \hookrightarrow n_2 \hookrightarrow c_2 \wedge$
   there exists $c_{nonxor} \in \{c_1, c_2\}, e_{nonxor} \in \{e_1, e_2\}$:
   $l(c_{nonxor}) \neq xor \ \wedge \ a_{nonxor} = (c_{nonxor}, e_{nonxor})$

Construction:

2) $E_2 = E_1 \setminus \{e_{nonxor}\}$
3) $F_2 = F_1$
4) $C_2 = C_1$
5) $A_2 = A_1 \setminus \{a_{nonxor}\}$
6) $I_{A_2} = I_{A_1} \setminus \{i \in I_{A_1} \mid f_{A_1}(i) = a_{nonxor}\} \wedge$

$\forall i \in I_{A_2} : f_{A_2}(i) = f_{A_1}(i)$

Error Cases:

7) The reduction rule always reports an error since c_{nonxor} produces repeatedly tokens on a_{nonxor}.

Definition 4.11 (Reduction of Unstructured Cyclic Start and End Components (b))**.** Let EPC_1 and EPC_2 be two relaxed syntactically correct EPCs with the respective index sets I_{A_1} and I_{A_2}, index and count functions f_{A_1}, f_{A_2}, ξ_1, and ξ_2. The pair $((EPC_1, I_{A_1}), (EPC_2, I_{A_2})) \in T_{d2b}$ if the following conditions hold for one start or one end event $e_1, e_2 \in E_1$ and a connector $c_1, c_2 \in C_1$ of EPC_1, and if EPC_2 can be constructed from EPC_1 as follows:

Condition:

1) there exists $c_1, c_2 \in C_1, e_1, e_2 \in E_1$ and $a_1, a_2, a_3 \in A_1$ such that
$e_1 \neq e_2 \ \wedge \ c_1 \neq c_2 \wedge$
$\bullet e_1 = \emptyset \ \wedge \ \bullet e_2 = \emptyset \wedge$
$a_1 = (e_1, c_1,), a_2 = (e_2, c_2), a_3 = (c_1, c_2) \in A_1 \ \wedge \ (c_2, c_1) \notin A_1 \wedge$
$\xi(a_1, I_{A_1}) = 1 \ \wedge \ \xi(a_2, I_{A_1}) = 1 \ \wedge \ \xi(a_3, I_{A_1}) = 1 \wedge$
$\exists n_1 \in N \setminus \{c_1\} : c_1 \hookrightarrow n_1 \hookrightarrow c_1 \wedge$
$\exists n_2 \in N \setminus \{c_2\} : c_2 \hookrightarrow n_2 \hookrightarrow c_2 \wedge$
there exists $c_{and} \in \{c_1, c_2\}, e_{nonxor} \in \{e_1, e_2\}$:
$l(c_{and}) = and \ \wedge \ a_{and} = (e_{and}, c_{and})$

Construction:

2) $E_2 = E_1 \setminus \{e_{and}\}$
3) $F_2 = F_1$
4) $C_2 = C_1$
5) $A_2 = A_1 \setminus \{a_{and}\}$
6) $I_{A_2} = I_{A_1} \setminus \{i \in I_{A_1} \mid f_{A_1}(i) = a_{and}\} \wedge$
$\forall i \in I_{A_2} : f_{A_2}(i) = f_{A_1}(i)$

Error Cases:

7) The reduction rule always reports an error since c_{and} will not have a token on its input arc a_{and} in the second execution of the cyclic part.

Both these rules for unstructured cyclic start and end components preserve the car-

dinality restrictions of all nodes and the coherence restriction of relaxed syntactical correctness. If we had considered only a single pair of a connector and an event we would not be able to guarantee that every node would still be on a path from a start to an end node.

Delta Components

In this section, we will discuss a subpart of an EPC built from three connectors that we call delta component. A delta component contains one input arc to a first split, which is followed by another split and a join in the postset. Furthermore, the preset of the join connector only includes the two splits. There are two output arcs from a delta component: one from the second split and one from the join. Basically, there are 3^3 types of delta components for each combination of three connector labels (see Figure 4.19). For some of the delta components, it is possible to eliminate the arc from the first split to the join or from the second split to the join. In some cases no reduction is possible. Furthermore, some delta components produce a deadlock or a contact situation at the join connector. If there is an error case an empty circle represents a missing token and a filled circle a token that is potentially too much (see Figures 4.20–4.22).

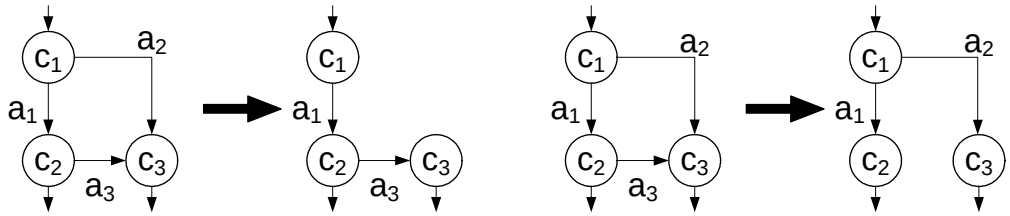


Figure 4.19: Reduction of Delta Components

Figure 4.20 illustrates delta components, where the first split is an XOR. If both split connectors are XORs (see first column), then the join should also be an XOR. In this case, the arc between the first split and the join can be deleted and still the same combination of outputs can be produced. This also holds if the second split is an OR. If the split is an AND, either both outputs, or only the right one is activated. Therefore, no arc can be deleted. The XOR delta component runs into a deadlock if the join is an AND, but should

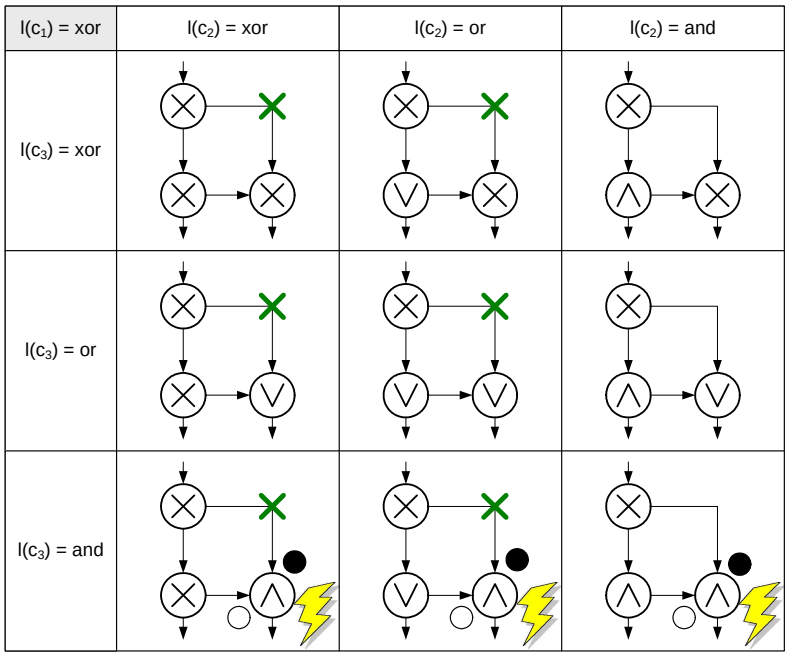


Figure 4.20: Reduction of XOR Delta Components

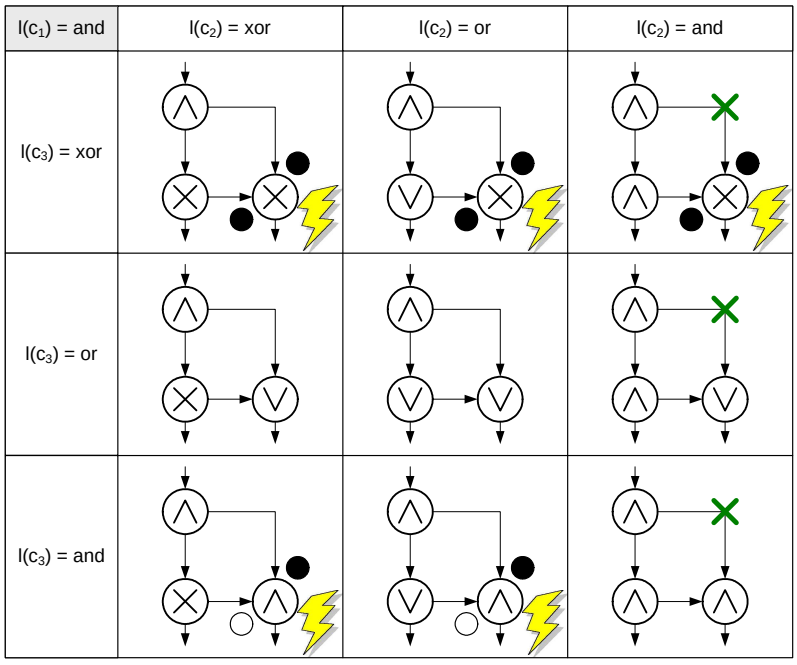


Figure 4.21: Reduction of AND Delta Components

be an XOR or an OR. Accordingly, we record the error and apply the reduction rule as if the connector had the appropriate label.

Figure 4.21 depicts delta components with an AND as the first split. In this case, a reduction is only possible if the second split is also an AND. The join should be an AND or an OR in order to avoid a lack of synchronization such as in the first row where the join is an XOR. This lack of synchronization results in contact situations in the reachability graph. If the join is an AND, but the second split is an XOR or an OR (third row), there is a potential deadlock since the arc between second split and join might not get a positive token. For both the first and the second column, no reduction can be applied. In this case, the left output is optional, while the right output is always covered.

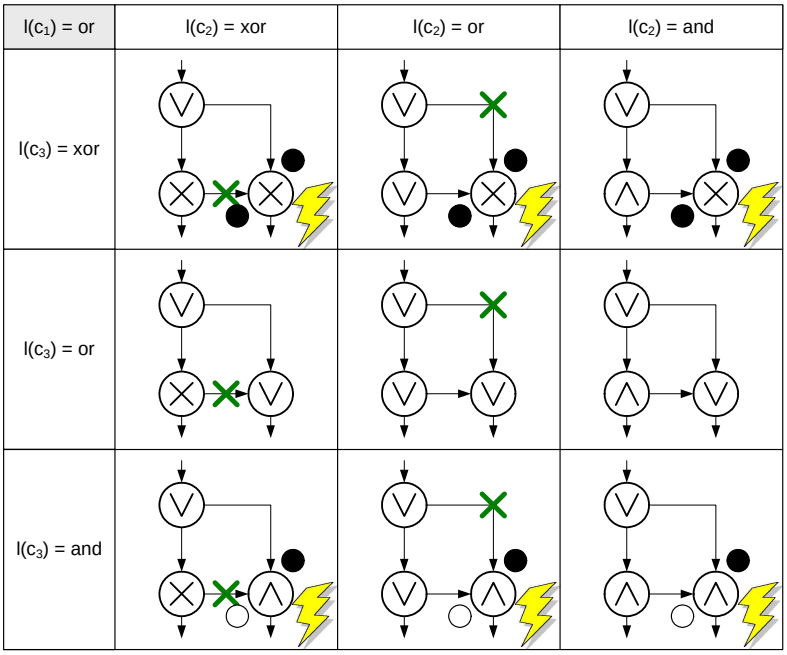


Figure 4.22: Reduction of OR Delta Components

Figure 4.22 shows those delta components that have an OR as first split connector. If the join is an XOR, there is a potential lack of synchronization with subsequent contact situations (first row). If the join is an AND, there might be tokens missing to fire (third row). The second row shows the well-behaving cases where the first OR-split is matched by an OR-join. In these cases, either one or both of the output arcs are taken. Accordingly,

the arc between the XOR-split and the OR-join (first column)[5], and the arc between the first OR-split and OR-join (second column) can be deleted without restricting the output combinations. If the second split is an AND, the left output is optional and the right is always covered. Therefore, no reduction can be applied.

Definition 4.12 (Reduction of Delta Components)**.** Let EPC_1 and EPC_2 be two relaxed syntactically correct EPCs with the respective index sets I_{A_1} and I_{A_2}, index and count functions f_{A_1}, f_{A_2}, ξ_1, and ξ_2. The pair $((EPC_1, I_{A_1}), (EPC_2, I_{A_2})) \in T_e$ if the following conditions hold for three connectors $c_1, c_2, c_3 \in C_1$, and if EPC_2 can be constructed from EPC_1 as follows:

Condition:

1) there exists $c_1, c_2, c_3 \in C_1, a_1, a_2, a_3 \in A_1$ such that
   $c_1 \neq c_2 \wedge c_1 \neq c_3 \wedge c_2 \neq c_3 \wedge$
   $a_1 = (c_1, c_2), a_2 = (c_1, c_3), a_3 = (c_2, c_3) \wedge$
   $c_1\bullet = \{c_2, c_3\} \ \wedge \ \bullet c_2 = \{c_1\} \ \wedge \ \bullet c_3 = \{c_1\} \wedge$
   $\xi(a_1, I_{A_1}) = 1 \ \wedge \ \xi(a_2, I_{A_1}) = 1 \ \wedge \ \xi(a_3, I_{A_1}) = 1 \wedge$

Construction for $l(c_1) = l(c_2) \vee (l(c_1) = xor \wedge l(c_2) = or)$*:*

2) $E_2 = E_1$
3) $F_2 = F_1$
4) $C_2 = C_1$
5) $A_2 = A_1 \setminus \{a_2\}$
6) $I_{A_2} = I_{A_1} \setminus \{i \in I_{A_1} \mid f_{A_1}(i) = a_2\}$
7) $\forall i \in I_{A_2} : f_{A_2}(i) = f_{A_1}(i)$

Construction for $l(c_1) = or \wedge l(c_2) = xor$*:*

8) $E_2 = E_1$
9) $F_2 = F_1$
10) $C_2 = C_1$
11) $A_2 = A_1 \setminus \{a_3\}$
12) $I_{A_2} = I_{A_1} \setminus \{i \in I_{A_1} \mid f_{A_1}(i) = a_3\} \wedge$
    $i \in I_{A_2} : f_{A_2}(i) = f_{A_1}(i)$

[5]Please note that this changes the behavior since the OR-join does not have to wait for the XOR-split. Still, the combination of tokens that can be produced on the remaining output arcs of the second and the third connector are the same as in the unreduced case.

Error cases:
13) $l(c_1) = xor \wedge l(c_3) = and$
14) $l(c_1) = and \wedge l(c_3) = xor$
15) $l(c_1) = and \wedge l(c_2) \neq and \wedge l(c_3) = and$
16) $l(c_1) = or \wedge l(c_3) = xor$
17) $l(c_1) = or \wedge l(c_3) = and$

The rule preserves both the cardinality restrictions of all nodes, and the coherence restriction of relaxed syntactical correctness.

Prism Components

In the previous section, we have seen four cases of delta components that could not be reduced: (1) c_1 is an XOR and c_2 is an AND, (2) c_1 is an AND and c_2 is an XOR, (3) c_1 is again an AND and c_2 an OR, and (4) c_1 is an OR and c_2 an AND. These cases have in common that one output arc is activated optionally, while the other always receives a positive token. Figure 4.23 shows that these delta components can be extended with a fourth connector c_4 to become a prism component. The four cases must always have an OR-join as a fourth connector in order to provide proper synchronization of the mandatory and the optional branch. If the fourth is not an OR, the model is not sound due to a potential deadlock (AND) or a contact situation (XOR). The prism can be reduced by deleting the arc a_3 between the second and the third connector. The type of the third connector is not considered here, since the delta rule already contributes reduction and error reports concerning the interplay of connectors c_1 to c_3.

Definition 4.13 (Reduction of Prism Components)**.** Let EPC_1 and EPC_2 be two relaxed syntactically correct EPCs with the respective index sets I_{A_1} and I_{A_2}, index and count functions f_{A_1}, f_{A_2}, ξ_1, and ξ_2. The pair $((EPC_1, I_{A_1}), (EPC_2, I_{A_2})) \in T_f$ if the following conditions hold for three connectors $c_1, c_2, c_3, c_4 \in C_1$, and if EPC_2 can be constructed from EPC_1 as follows:
Condition:

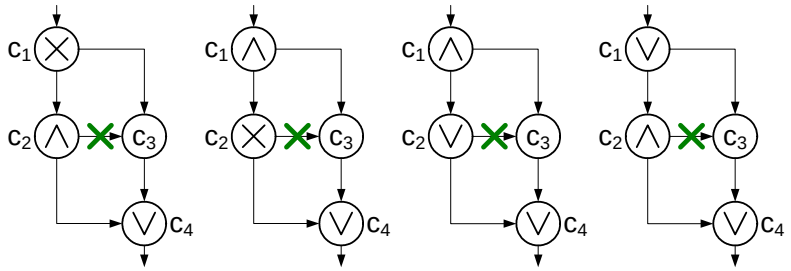


Figure 4.23: Reduction of Prism Components

1) there exists $c_1, c_2, c_3, c_4 \in C_1, a_1, a_2, a_3, a_4, a_5 \in A_1$ such that
$c_1 \neq c_2 \wedge c_1 \neq c_3 \wedge c_1 \neq c_4 \wedge c_2 \neq c_3 \wedge c_2 \neq c_4 \wedge c_3 \neq c_4 \wedge$
$((l(c_1) = xor \wedge l(c_2) = and) \vee (l(c_1) = and \wedge l(c_2) = xor) \vee$
$(l(c_1) = and \wedge l(c_2) = or) \vee (l(c_1) = or \wedge l(c_2) = and)) \wedge$
$a_1 = (c_1, c_2), a_2 = (c_1, c_3), a_3 = (c_2, c_3), a_4 = (c_2, c_4), a_5 = (c_3, c_4) \wedge$
$c_1\bullet = \{c_2, c_3\} \wedge \bullet c_2 = \{c_1\} \wedge \bullet c_3 = \{c_1\} \wedge$
$c_2\bullet = \{c_3, c_4\} \wedge c_3\bullet = \{c_4\} \wedge \bullet c_4 = \{c_2, c_3\} \wedge$
$\xi(a_1, I_{A_1}) = 1 \wedge \ldots \wedge \xi(a_5, I_{A_1}) = 1$

Construction:

2) $E_2 = E_1$
3) $F_2 = F_1$
4) $C_2 = C_1$
5) $A_2 = A_1 \setminus \{a_3\}$
6) $I_{A_2} = I_{A_1} \setminus \{i \in I_{A_1} \mid f_{A_1}(i) = a_3\} \wedge$
$\forall i \in I_{A_2} : f_{A_2}(i) = f_{A_1}(i)$

Error cases:

7) $l(c_4) \neq or$

The rule preserves both the cardinality restrictions of all nodes and the coherence restriction of relaxed syntactical correctness.

Connector Merge

Van Dongen et al. point to the fact that two consecutive joins or splits of the same connector type provide the same behavior as if both were merged. Figure 4.24 illustrates the respective reduction rules. One consequence of such a merger is that the identity of the individual connectors is lost. This might be a problem for errors that are found by further rules, since it is not clear which of the merged connectors is responsible for the error.

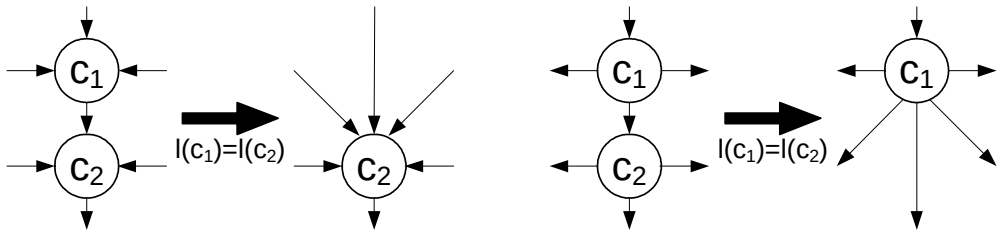


Figure 4.24: Connector Merge

Definition 4.14 (Connector Merge)**.** Let EPC_1 and EPC_2 be two EPCs. The pair $(EPC_1, EPC_2) \in T_g$ if the following conditions hold for two connectors $c_1, c_2 \in C_1$, and if EPC_2 can be constructed from EPC_1 as follows:

Condition:

1) there exists $(c_1, c_2 \in J_1 \wedge (c_1, c_2) \in A_1) \vee (c_1, c_2 \in S_1 \wedge (c_1, c_2) \in A_1)$ such that
   $c_1 \neq c_2 \wedge$
   $(c_2, c_1) \notin A_1 \wedge$
   $l(c_1) = l(c_2) \wedge$
   $\xi((c_1, c_2), I_{A_1}) = 1$

Construction for $c_1, c_2 \in J_1$*:*

2) $E_2 = E_1$
3) $F_2 = F_1$
4) $C_2 = C_1 \setminus \{c_1\}$
5) $A_2 = \{(x, y) \in A_1 \mid x \neq c_1 \wedge y \neq c_1\} \cup \{(x, y) | x \in {\bullet}c_1 \wedge y = c_2\}$
6) $I_{A_2} = \{i \in I_{A_1} \mid f_{A_1}(i) \neq (c_1, c_2)\} \wedge$
   $\forall\{i \in I_{A_2} \wedge (x, y) = f_{A_1}(i)\} : f_{A_2}(i) = \begin{cases} (x, y) \text{ if and only if } x \neq c_1 \wedge y \neq c_1 \\ (x, c_2) \text{ if and only if } y = c_1 \end{cases}$

Construction for $c_1, c_2 \in S_1$*:*
6) $E_2 = E_1$
7) $F_2 = F_1$
8) $C_2 = C_1 \setminus \{c_2\}$
5) $A_2 = \{(x,y) \in A_1 \mid x \neq c_2 \wedge y \neq c_2\} \cup \{(x,y) | x = c_1 \wedge y \in c_2\bullet\}$
6) $I_{A_2} = \{i \in I_{A_1} \mid f_{A_1}(i) \neq (c_1, c_2)\} \wedge$
$\forall\{i \in I_{A_2} \ \wedge (x,y) = f_{A_1}(i)\} : f_{A_2}(i) = \begin{cases} (x,y) \text{ if and only if } x \neq c_1 \wedge y \neq c_1 \\ (c_1,y) \text{ if and only if } x = c_2 \end{cases}$
There are no error cases.

The rule preserves both the cardinality restrictions of all nodes and the coherence restriction of relaxed syntactical correctness. No self-arcs can be created since $(c_2, c_1) \notin A_1$.

Homogeneous

Similar to the Petri net class of state machines (see e.g. [Des05, p.172]), EPCs with no other than XOR-connectors are trivially correct. Consider the set of initial markings, which includes for each start arc an initial marking, where it is the only arc having a positive token. Since there are only XOR-connectors, there is no deadlock and the sum of positive tokens is one in all markings. Furthermore, due to the relaxed syntactical correctness of the EPC, every node is on a path from a start to an end node. This also facilitates that all end arcs are reached from the described set of initial markings. Therefore, an EPC with XOR-connectors only is sound. Furthermore, acyclic EPCs are also correct, if either there are only OR-connectors or if there are only AND-connectors and OR-joins. These cases are similar to the flow activity in BPEL which is correct by design (cf. [VA06]).

Definition 4.15 (Reduction of Homogeneous EPCs)**.** Let EPC_1 and EPC_2 be two EPCs. The pair $(EPC_1, EPC_2) \in T_h$ if the following conditions hold for C_1, and if EPC_2 can be constructed from EPC_1 as follows:
Condition:
1) $\forall c \in C_1 : l(c) = xor \vee$

$(\nexists n \in N : n \hookrightarrow n \ \wedge \ \forall c \in C_1 : l(c) = or) \vee$

$(\nexists n \in N : n \hookrightarrow n \ \wedge \ \forall c \in S_1 : l(c) = and \ \wedge c \in J_1 : l(c) \neq xor)$

Construction:

2) $E_2 = \{e_1, e_2\}$

3) $F_2 = \emptyset$

4) $C_2 = \emptyset$

5) $A_2 = \{(e_1, e_2)\}$

6) $I_{A_2} = \{1, 2\}$

7) $f_{A_2}(1) = e_1, f_{A_2}(2) = e_2$

There are no error cases.

The homogeneous rule is the desirable last reduction. It obviously preserves relaxed syntactical correctness, and it yields the trivial EPC, which implies that there are not more errors in the source EPC_1. After applying this rule, it is easy to determine whether the original EPC was sound. If there are errors recorded in the reduction process, the EPC is not sound. Otherwise, it is sound.

4.3.3 A Reduction Algorithm for EPCs

In the previous section, we have defined a set of reduction rules for the verification of EPC soundness. Algorithm 3 illustrates through object-oriented pseudo code how the rules can be prioritized. Each invocation of a rule that has error cases takes the error list to append further errors (lines 7-11). The algorithm tries to minimize the utilization of the connector merge rule, since losing the identity of a connector poses problems in finding the connector that is responsible for an error. Accordingly, the inner while-loop (lines 3-13) calls all reduction rules except the connector merge as long as the EPC can be reduced. If that is no longer possible, i.e. the new number of arcs and nodes equal the old values, the loop is exited and the algorithm tries to merge one connector as part of the outer while-loop. If that succeeds, the inner loop is reentered again, otherwise the reduction terminates. As a result, the size of the reduced EPC and a list of errors is returned.

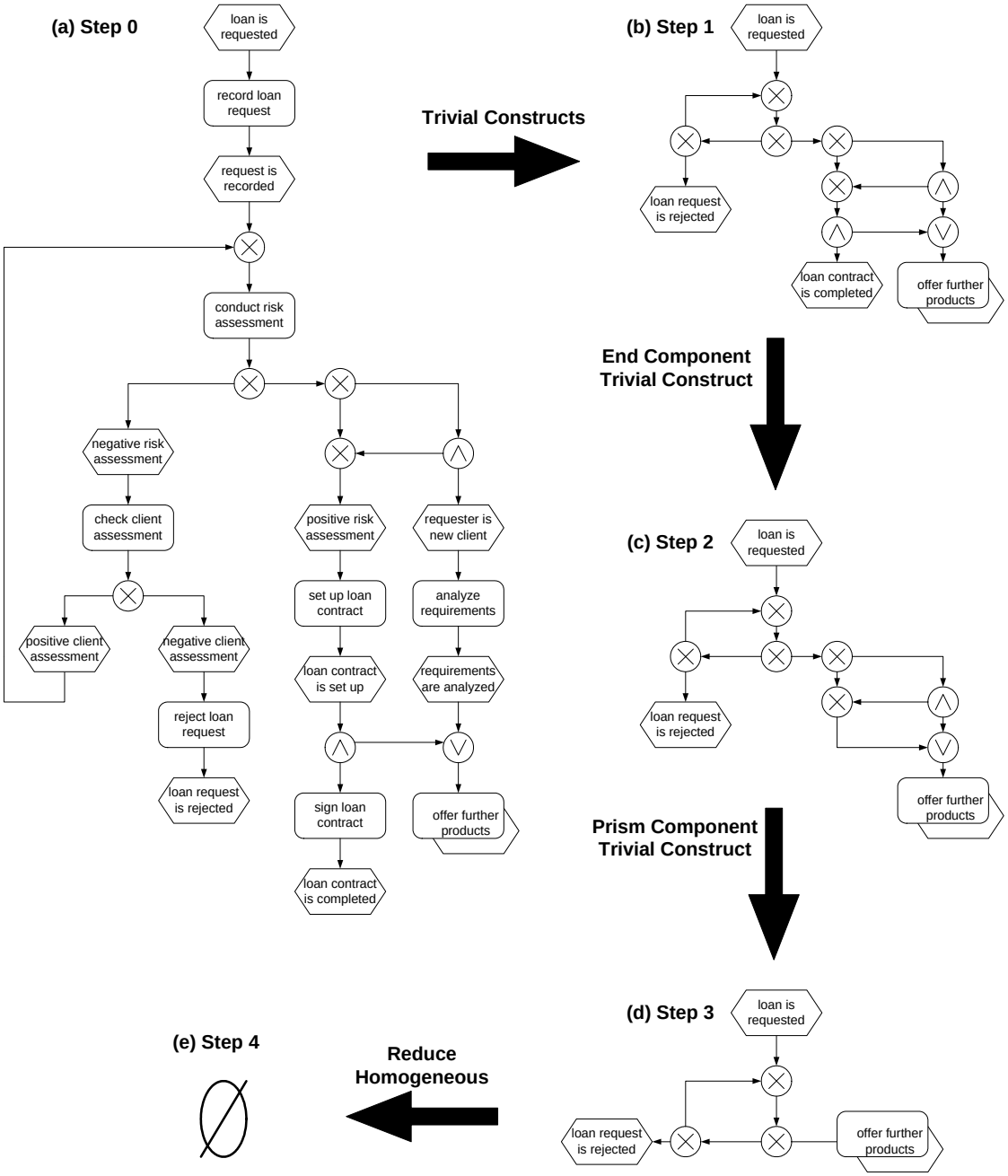


Figure 4.25: Stepwise reduction of the Loan Request EPC

Algorithm 3 Pseudo code for reduction of an EPC

Require: EPC

1: $nodes \leftarrow |N|, arcs \leftarrow |A|, nodesnew \leftarrow 0, arcsnew \leftarrow 0, ErrorList \leftarrow \emptyset$
2: **while** $nodes \neq nodesnew \vee arcs \neq arcsnew$ **do**
3: **while** $nodes \neq nodesnew \vee arcs \neq arcsnew$ **do**
4: $nodes \leftarrow |N|, arcs \leftarrow |A|$
5: $reduceHomogeneous(EPC)$
6: $reduceTrivialConstructs(EPC)$
7: $reduceStructuredBlocks(EPC, ErrorList)$
8: $reduceStructuredLoop(EPC, ErrorList)$
9: $reduceStartEndComponents(EPC, ErrorList)$
10: $reduceDeltaComponent(EPC, ErrorList)$
11: $reducePrismComponent(EPC, ErrorList)$
12: $nodesnew \leftarrow |N|, arcsnew \leftarrow |A|$
13: **end while**
14: $mergeConnector(EPC)$
15: $nodesnew \leftarrow |N|, arcsnew \leftarrow |A|$
16: **end while**
17: **return** $|N|, ErrorList$

Figure 4.25 illustrates how the reduction algorithm works on the Loan Request EPC from page 38. Deleting the *trivial constructs* yields the EPC that is shown in (b). It consists of eight connectors, one start event, two end events, and an end process interface. Applying the *end component reduction* and the trivial construct reduction results in the EPC depicted in (c). On the right-hand side, there is a delta component that cannot be reduced, but which, together with the OR-join as a fourth connector, yields a well-behaving prism component. The *prism component reduction* then results in the EPC given in (d). Since there are only XOR-connectors left, the Homogeneous Reduction can be applied to reduce the EPC to the trivial EPC.

Algorithm 3 was implemented in a batch program called *xoEPC*.[6] *xoEPC* is written in the object-oriented scripting language XOTcl [NZ00], which is an extension of Tcl (see [WJH03]). Figure 4.26 gives an overview of input files that are read by *xoEPC* and the output files generated by it. *xoEPC* loads all *.xml files from the current directory

[6] Some of the functionality of *xoEPC* is available via a web interface at http://wi.wu-wien.ac.at/epc.

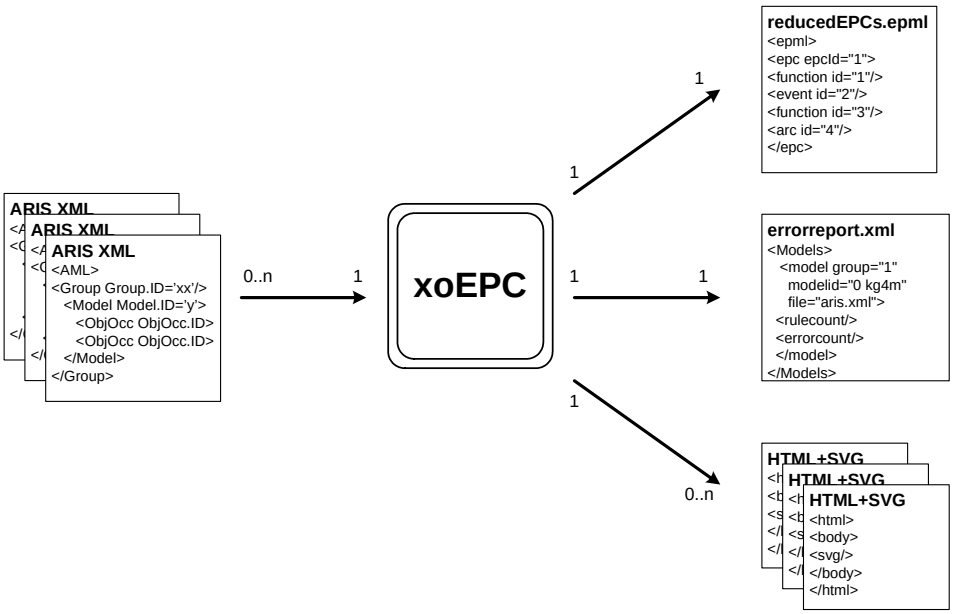


Figure 4.26: xoEPC inputs and outputs

and checks whether they are ARIS XML files [IDS01, IDS03b]. If they are, the XML is processed with the tDOM package [Löw00], a Tcl implementation of the DOM specification[7]. For each EPC model that has at least one event and one function, *xoEPC* checks relaxed syntactical correctness and applies the reduction algorithm. The internal data structure of xoEPC uses an adjacency matrix representation of the EPC and the reduction methods work on this data structure. All EPCs that cannot be reduced completely are written to the reducedEPCs.epml file.[8] These EPCs can be further analyzed by loading them into ProM via the EPML import. If errors are encountered they are recorded in the errorresults.xml file. This file also records the processing time of the reduction, metadata of the model, as well as the size of the original and the size of the reduced EPC. Finally, an XHTML file with an embedded SVG graphic[9] is generated for each EPC

[7]For an overview of the various DOM specifications of the World Wide Web Consortium refer to http://www.w3.org/DOM/DOMTR.

[8]The reason for using EPML as an output format and not the ARIS file format is that the EPML representation is more compact and easier to generate. For details refer to [MN04b, BHK+06]. The ARIS format is chosen as an input format since many EPCs such as those of the SAP Reference Model are available in ARIS.

[9]For the respective specifications, see XHTML [Pem02] and SVG [FJJ03].

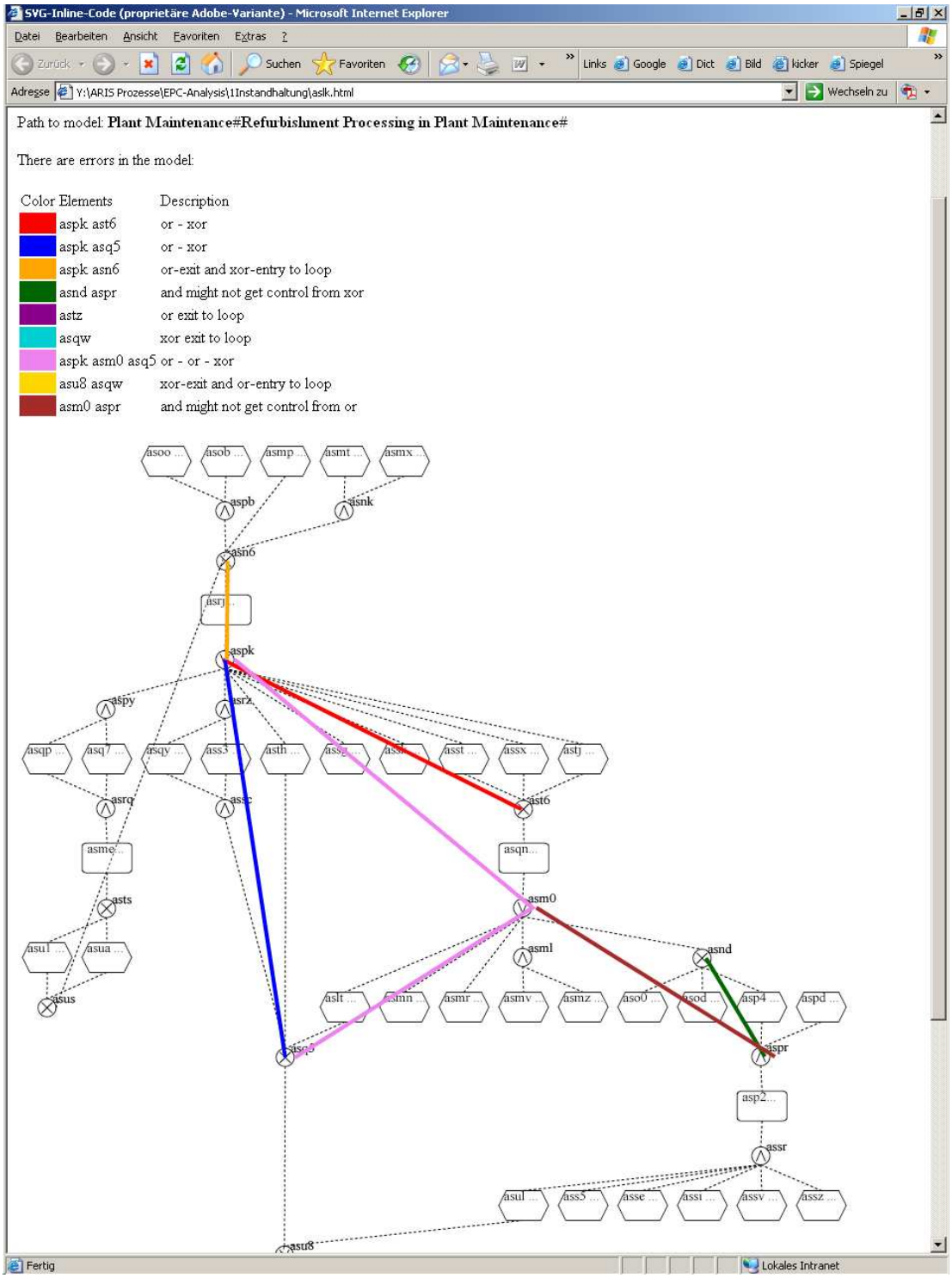


Figure 4.27: Snapshot of an SVG generated by xoEPC for the Plant Maintenance – Refurbishment Processing in Plant Maintenance EPC

based on the position information in the ARIS XML file. It projects the errors back to the model by highlighting the involved connectors. Figure 4.27 shows an example of an XHTML+SVG file generated by *xoEPC*. The different colors refer to the errors that are listed at the top of the screen. We discuss the errors in detail on page 153. Using this visual information the modeler can easily fix the problems.

4.3.4 Reduction of the SAP Reference Model

In order to test the performance of the reduction kit, we turn back to the SAP Reference Model [KT98] that was mentioned before. This extensive model collection includes almost 10,000 individual models. 604 of them are EPCs that have at least one event and one function and 600 of them are interpretable, i.e. they do not include functions and events with more that one input or output arc. On the following pages, we discuss the reduction performance from four perspectives: (1) Processing time of reduction, (2) Extent of reduction, (3) Applicability of reduction rules, and (4) Number of errors found.

Processing Time of Reduction

The processing of the whole SAP Reference Model took about 18 minutes on a desktop computer with a 3.2 GHz processor. Figure 4.28 shows how the processing time is distributed over different models. Each EPC model is represented by a number on the horizontal axis ordered by the processing time, i.e. the EPC that was processed the fastest is on the very left position, and the most time-consuming EPC is found on the very right-hand side of the spectrum. Each EPC is assigned its processing time as the ordinate. Please note that the ordinate is given in a logarithmic scale. In Figure 4.28, it can be seen that about 320 models are processed in less that 1,000 milliseconds, i.e. one second. Furthermore, we see that only a few (16 EPCs) take more than 10 seconds. It is interesting to note that the 13 largest models with more than 80 nodes are also the 13 models that require the most processing time. The number of nodes and the processing time are correlated with a Pearson's coefficient of correlation of 0.592, showing that the performance very much depends on the size of the EPC. The maximum processing time

was two minutes and 22 seconds for an EPC with 111 nodes, 136 arcs, and 32 connectors. The performance of the analysis is much better than the performance of the relaxed soundness analysis reported in [MMN+06b, MMN+06c], which took about seven hours and 45 minutes on the same computer, in contrast to less than 30 minutes with reduction rules.

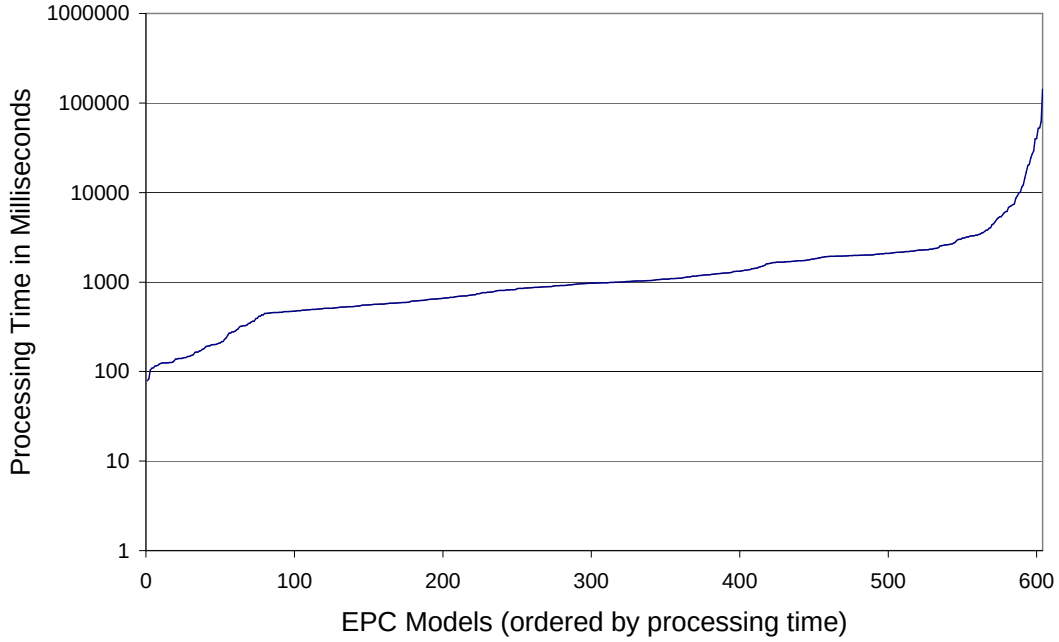


Figure 4.28: Processing time for reducing an EPC

Extent of Reduction

In this section, we will discuss the extent of the reduction by comparing the number of nodes from before and after the reduction. All original EPCs together include 12,529 nodes, while the reduced EPCs have only 1,118 altogether. The average per model is about 21 nodes before and 1.85 nodes after the reduction. This means that 91% of the nodes are deleted in the various reduction steps. Figure 4.29 shows the EPCs ordered by size of the reduced model related to the reduced size. Two things can be seen from this figure. Firstly, 103 of 604 EPCs could not be reduced completely. Furthermore, these

103 EPCs have less than 29 nodes, which is a bit more than what is the average for the unreduced models. Only 15 of them have more than 15 nodes.

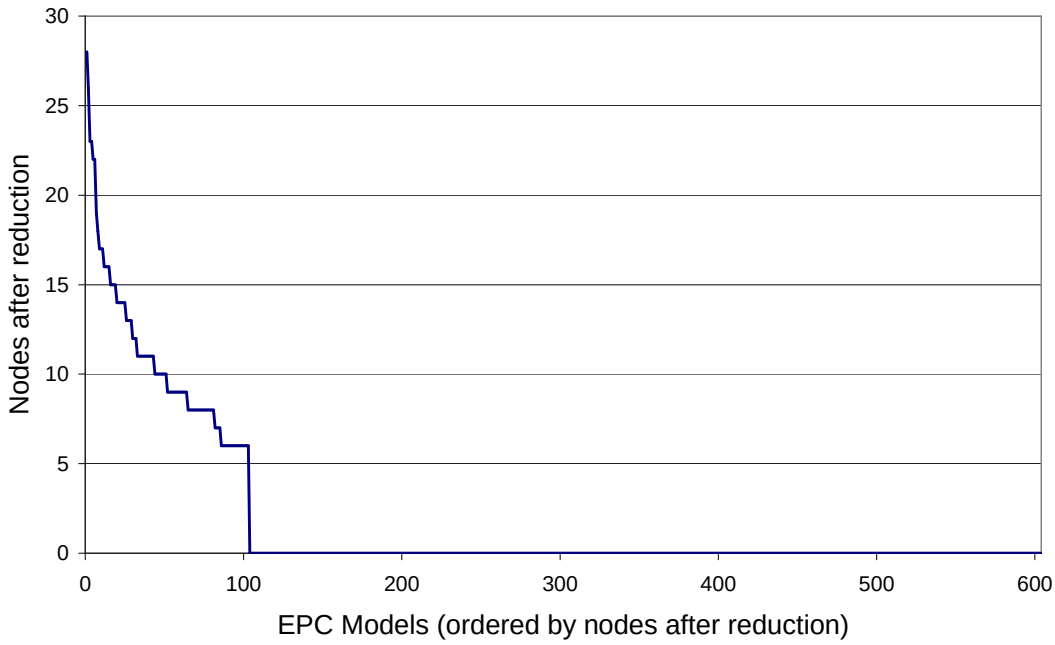


Figure 4.29: Size of reduced EPCs

Applicability of Reduction Rules

Figure 4.30 shows how often the eight reduction rules can be applied for the verification for the SAP Reference Model. Not surprisingly, the reduction of trivial constructs is used most often (about 6,600 times). It is followed by the start and end component reduction with about 2,400 applications. This is a good result, considering that large sets of start and end events have been a major verification problem in previous studies (see [VA06, MMN+06c]). The structured block rule is the third best regarding the frequency of application with 345 reductions. The homogeneous rule is still applied quite often (460 times), since this rule can only be applied once for an EPC. The remaining four rules are applied less frequently. They still play an important role for the reduction approach as a

whole. Running the reduction without the structured loop, delta, prism, and merge rule causes 13 additional models to not be reduced completely, and 53 errors would be missed.

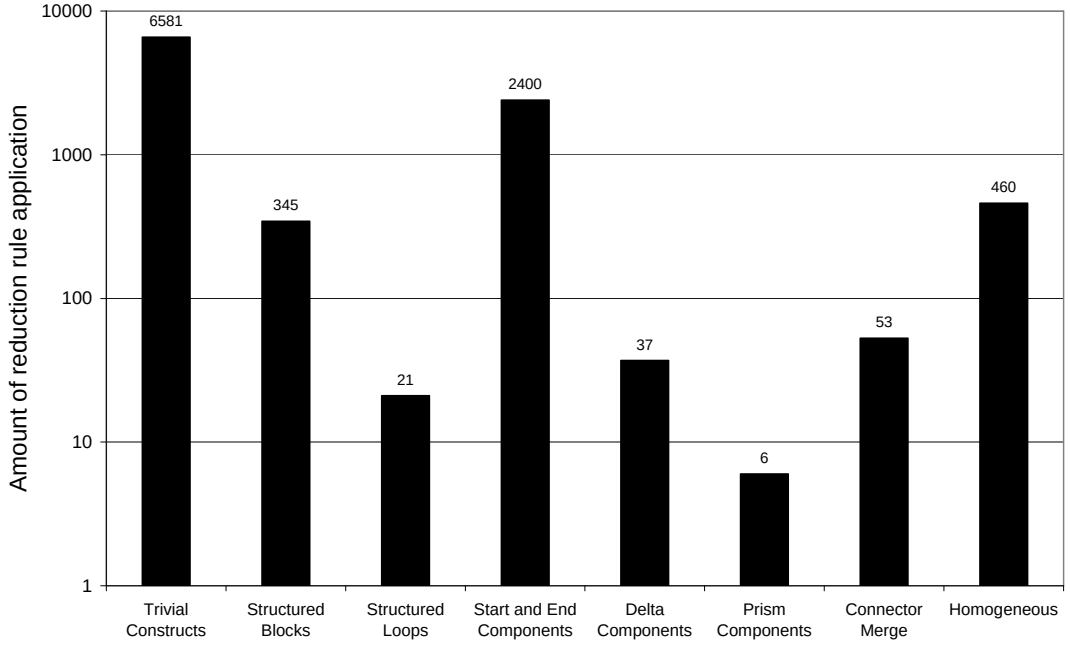


Figure 4.30: Number of Reduction Rule Applications (on logarithm scale)

Some of the reduced EPCs yield a specific pattern that could be used for designing additional reduction rules. The Figures 4.31–4.34 show four patterns that were found multiple times in the set of reduced EPCs. The reachability graph analysis with ProM can be used to verify them. While the EPCs of Figures 4.31, 4.32, and 4.34 are not sound, the Figure 4.33 is correct. Analyzing such reduced EPCs systematically is subject of future research.

Number of Errors found

Prior research on the verification of the EPCs of the SAP Reference Model has shown that there are several formal errors in the models (cf. [ZR96, DAV05, DJV05, MMN+06b]). In [MMN+06b], the authors identify a lower bound for the number of errors of 34 (5.6%)

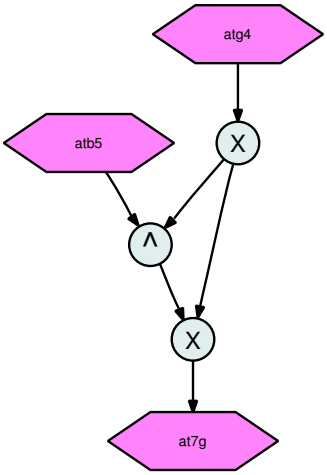


Figure 4.31: Two input component (not sound)

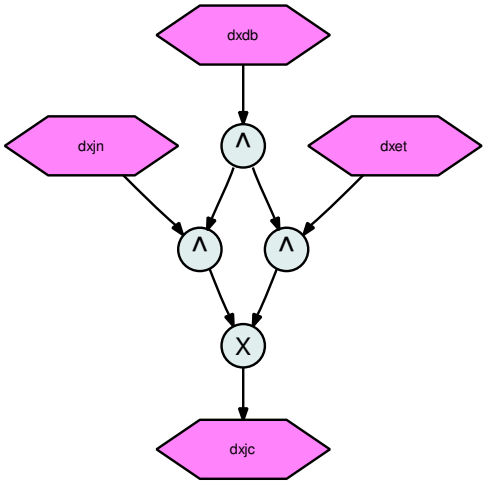


Figure 4.32: Three input component (not sound)

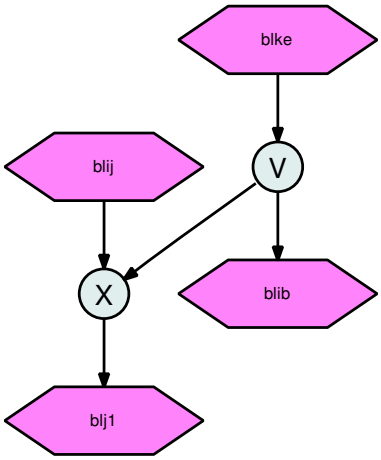


Figure 4.33: Acyclic two input two output component (sound)

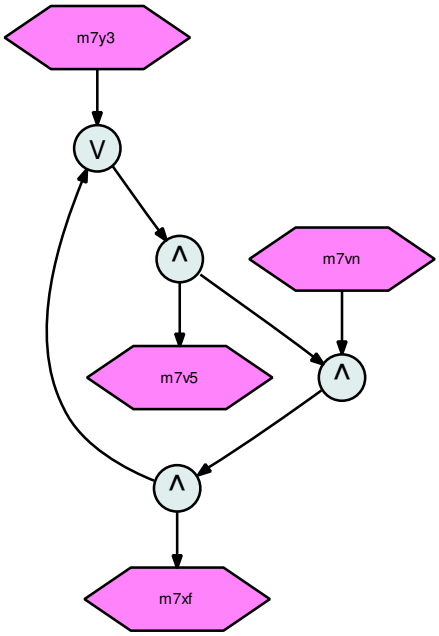


Figure 4.34: Cyclic two input two output component (not sound)

using the relaxed soundness criterion. The reduction based on *xoEPC* identifies 90 models for which altogether 178 error patterns were found (see Figure 4.35). This is almost three times as many models as in an earlier study which used the relaxed soundness criterion (see [MMN+06b]). Furthermore, there are 103 models that are not completely reduced, and for 57 of them no error was found in the reduction phase. We analyzed these models using the reachability graph approach. 68 of the 103 unreduced EPCs were not sound, and 36 unsound EPCs were not detected by the reduction rules. This yields 126 EPCs with errors in total for the SAP Reference Model.

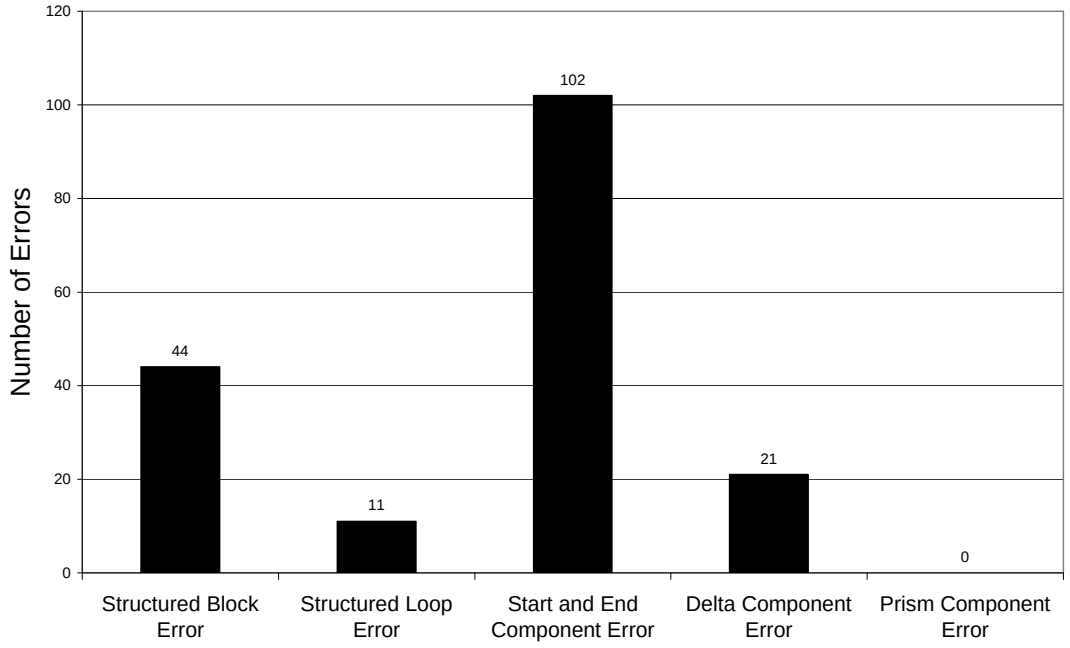


Figure 4.35: Number of Errors per Type

Comparing the application of a rule to the error cases, as depicted in Figure 4.35, offers some first inside regarding the question why errors are introduced in models. While 44 errors in structured blocks is often in absolute terms, it is little compared to the 345 times a structured block was reduced. This is different for structured loops and delta components: the relation of errors to no-error cases is about 1:2, i.e. 11 to 21 for loops and 21 to 37 for deltas. It seems as if modelers have more problems with these sophisticated building blocks than with the structured blocks. The start and end components may be

regarded as the extreme opposite, with 102 error cases compared to 2,400 applications. Still, this is the highest value in absolute terms and points to a problem with using unstructured start and end events. Surprisingly, the prism is applied six times. Even though it is the reduction pattern with the most nodes, it causes no errors. One explanation could be that modelers are aware that non-trivial components are best joined with an OR.

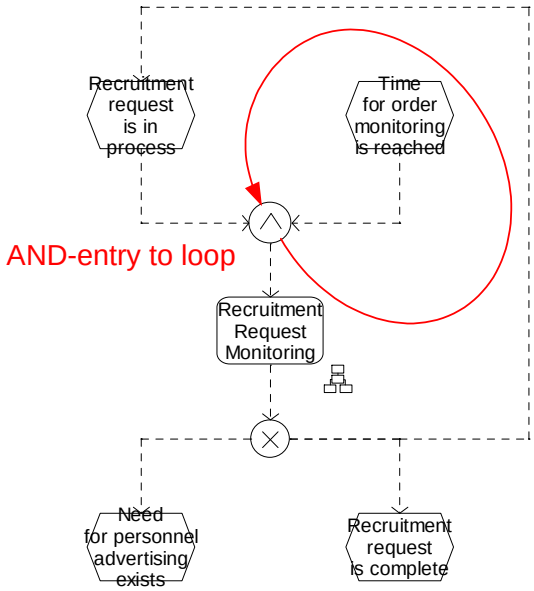


Figure 4.36: Recruitment – Recruitment – Recruitment Request Monitoring

Figure 4.36 depicts one of the small EPCs from the SAP Reference Model that was identified to be unsound by the reduction algorithm. *xoEPC* reports that there is a loop with an AND-join as entry and an XOR-split as exit. Obviously, the only possible initial marking including *Time for order monitoring is reached* cannot lead to a proper execution of the process, since the AND-join cannot receive a token on its second input arc. Figure 4.37 shows another example EPC for which nine errors were found by *xoEPC*. Errors 1 and 2 are mismatches of an OR-split with XOR-join connectors. These components might potentially create more than one token on subsequent arcs. The third error is an OR-exit to a loop. If this loop is executed multiple times, it is possible that multiple tokens are created at the exit. Error 4 relates to an AND-join that might not get control

from a previous XOR-split. This implies that there is no initial marking in which a positive token on the start arc pointing to the AND-join is guaranteed to run into a proper completion. The same holds true for error 9. Errors 5,6, and 8 relate to the loop at the bottom with its OR-entry and AND-exits and OR-exits. Error 7 is a delta component with two OR-splits and one XOR-join. Again, there is a potential lack of synchronization at the XOR-join. Furthermore, the reduction rules could not reduce the EPC completely. An analysis with ProM reveals that the AND-join with the start arc involved in error 4 and 9 might also not get control from the OR-split behind the first function *Order*. All unsound models that were detected by *xoEPC* are listed in the Appendix A. EPCs that were not completely reduced and checked with the ProM plug-in are depicted in Appendix B.

4.4 Summary

In this section, we presented an EPC-specific version of soundness as a correctness criterion for EPCs. We proposed two different approaches for soundness verification, one based on the reachability graph and a second based on reduction rules. While the first approach explicitly considers all markings and transitions of an EPC, there is a state explosion problem, as the maximum number of markings grows exponentially with the number of arcs. In order to avoid a performance problem, we introduced a set of reduction rules. This set extends prior work with new reductions for start and end components, delta components, prism components, and homogeneous EPCs. Both approaches were implemented as a proof-of-concept, the reachability graph verification approach as a plug-in within the ProM framework, and the reduction rules as a batch program called *xoEPC* written in XOTcl. Furthermore, we tested the performance of *xoEPC* in the verification of the SAP Reference model, which showed that the reduction rules approach is *fast*, that it provides a *precise* result for almost all models, and that it finds *three times as many errors* as other approaches based on relaxed soundness. In the following chapter, we discuss which model attributes could have an impact on the error probability from a theoretical point of view. The elaborations provide a foundation for answering the question whether errors are introduced randomly in a model, or whether there are factors that influence error probability.

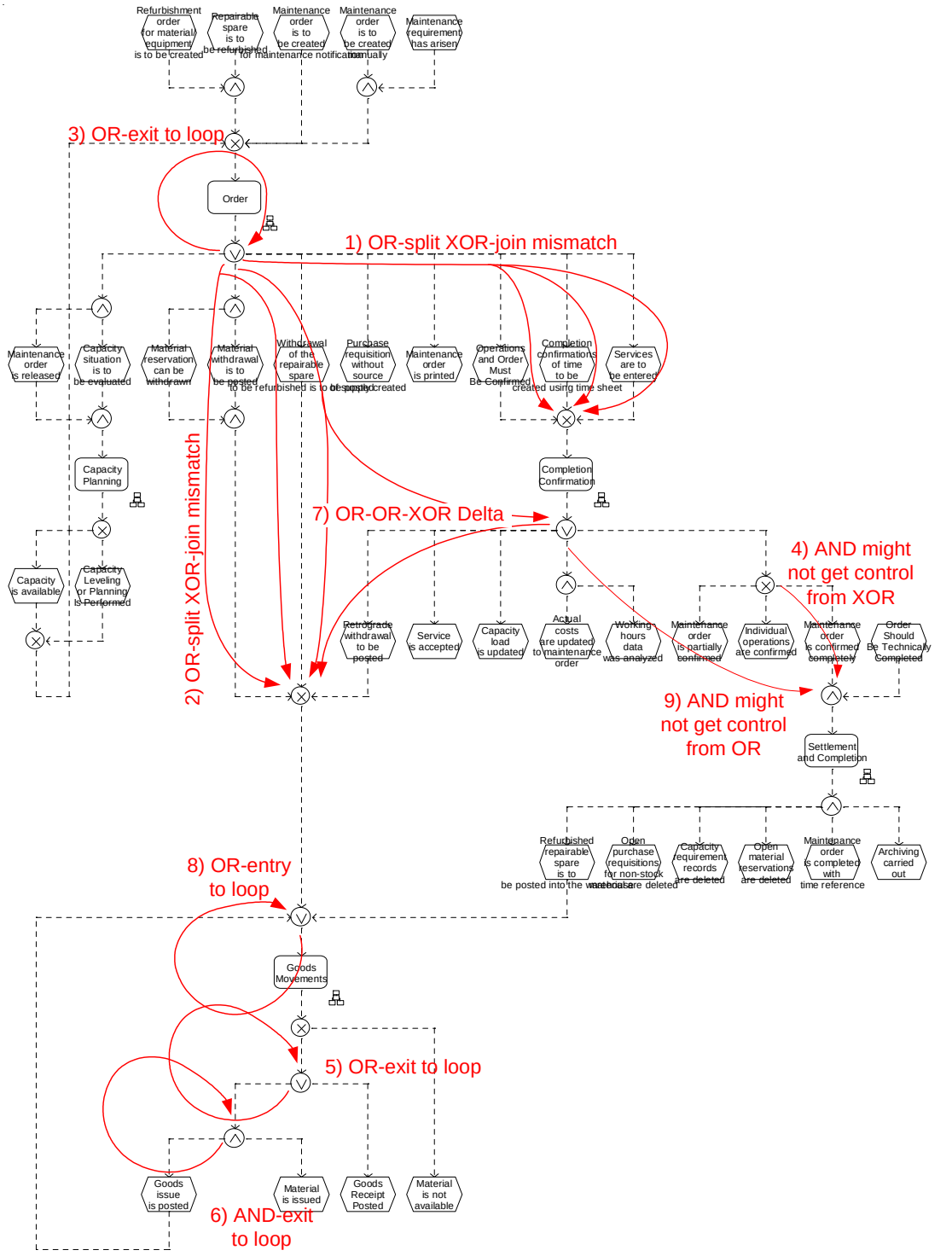


Figure 4.37: Plant Maintenance – Refurbishment Processing in Plant Maintenance

Chapter 5

Metrics for Business Process Models

Up to now, there has been little research on why people introduce errors in business process models in practice. In a more general context, Simon [Sim96] points to the limitations of cognitive capabilities and concludes that humans act rationally only to a certain extent. Concerning modeling errors, this argument would imply that human modelers lose track of the interrelations of large and complex models due to their limited cognitive capabilities, and then introduce errors that they would not insert in a small model. A recent study by *Mendling et al.* [MMN+06b] explores in how far certain complexity metrics of business process models have the potential to serve as error determinants. The authors conclude that complexity indeed appears to have an impact on error probability. Before we can test such a hypothesis in a more general setting, we have to establish an understanding of how we can define determinants that drive error probability and how we can measure them.

In this context, measurement refers to a process that assigns numbers or symbols to attributes of entities in the real world [FP97] in order to represent the amount or degree of those attributes possessed by the entities [Tor58, p.19]. This way, measurement opens

abstract concepts to an empirical evaluation and is therefore a cornerstone of natural, social, and engineering sciences. Related to this definition, an attribute refers to a property or a feature of an entity, while an entity may be an object or an event in the real world. Measurement at least serves the following three purposes: understanding, control, and improvement. The classical statement attributed to *Galilei* of "What is not measurable make measurable" stresses the ability of a measurement to deliver *understanding*. The principle idea behind this phrase is that measurement makes concepts more visible. In effect, entities and their relationships can be tracked more precisely, bringing forth a better understanding. In an emerging discipline like complexity of business process models, it might not be clear what to measure in the first place. Still, proposing and discussing measures opens a debate that ultimately leads to a greater understanding [FP97, p.7]. Secondly, measurement enables *control* in order to meet goals. According to *DeMarco*, "you cannot control what you cannot measure" [DeM82]. Based on an understanding of relationships between different attributes, one can make predictions, such as whether goals will be met and what actions need to be taken. For business process modeling projects it is important to establish suitable measurements since, as *Gilb* puts it, projects without clear goals will not achieve their goals clearly [Gil88]. The lack of measurements that can be automatically calculated from a give process model is a central problem of several quality frameworks, like the Guidelines of Modeling (GoM) [BRS95, SR98, BRU00], SEQUAL [LSS94, MSBS02, KSJ06], or the work of *Güceglioglu and Demirörs* [GD05b, GD05a]. While various empirical research has been conducted on quality aspects of data models (see e.g. [Moo01, GW03, Moo03, GW04, GW05]), such work is mostly missing for business process models [Moo05]. Defining quality concepts in a measurable way would be a major step towards understanding bad process design in general. Thirdly, measurement is crucial for the *improvement* of both business process models as products and business process modeling processes. In business science, it is an agreed upon insight from *Taylor's* scientific management [Tay11] to *Kaplan and Norton's* balanced scorecard [KN92, KN00] that measurement directs human behavior towards a goal. In organizational theory, this phenomenon was first recognized in the Hawthorne plant of Western Electric and is referred to as the Hawthorne Effect: what you measure will be improved (see e.g. [LB06, p,21]). Business process modeling has not yet established a general suite of measurements that is commonly accepted; therefore the potential for

improvements in current modeling practices is difficult to assess. This chapter aims to contribute to a more quantitatively oriented approach to business process modeling as we propose a set of potential error determinants for EPC business process models. This is also a step towards establishing business process modeling as an engineering discipline since "to predict and control effectively you must be able to measure. To understand and evaluate quality, you must be able to measure." [LB06, p.4].

Against this background, the remainder of this chapter is structured as follows. Section 5.1 presents the theoretical background of measurement with a focus on scale types and issues related to validity and reliability. Section 5.2 discusses which concepts are measured in the neighboring discipline of network analysis. We focus on degree, density, centrality, and connectivity metrics, since they seem to be promising for business process models. Section 5.3 gives an overview of complexity metrics in software engineering. We highlight the most prominent metrics and discuss their relationship to more abstract quality concepts for software products. In Section 5.4, we present related work on metrics for business process models. In Section 5.5, we identify the complexity of a process model's structure and its state space as the key determinants for error probability. Related to these two aspects we define a set of metrics and discuss their impact on error probability. Finally, Section 5.7 gives a summary before the metrics are tested in the subsequent chapter.

5.1 Measurement Theory

The representational theory of measurement (or measurement theory) explains how empirical phenomena can be captured in terms of a measurement (see [Zus91, FP97, SA05]). Formally, a measurement can be defined as a mapping (also called *scale*) from the domain of the empirical world to the range of mathematical concepts (see e.g. [FP97]). In software engineering, the terms metric and measurement are often used interchangeably. There actually seems to be a certain reluctance to make a clear distinction between both terms, partially to avoid confusion with the term metric in a mathematical sense (cf. [FP97, p.103]). Several software engineering books only give vague characteristics of a metric (see [GJM03, p.368], [Som01, p.567], or [LB06, p.9]). Throughout the remainder

of this thesis, we will use the term *metric* in order to refer to the *type* of a measurement that quantifies a certain attribute of an entity in the real world by following a predefined way of quantification. An implication of this definition is that entities of the real world can be compared by comparing the measurement for a certain metric. The term *statistic* is related to such an understanding of a metric since it refers to a sample as a specific entity of the real world [AS02, p.212]. A sample can be defined as a subset of measurements selected from a population while a population consists of the set of all measurements in which an investigator is interested [AS02, p.17].

Related to measurement, three problems have to be considered (see [VW93]). Firstly, the *representation problem* relates to the condition that the mapping should preserve relations that exist in the real world. Secondly, the *uniqueness problem* refers to invariance of a measurement under certain transformations. Thirdly, the *meaningfulness problem* is concerned with drawing conclusions about the truth of statements based on comparison of assigned scale values. While the third aspect is subject to ongoing research, the first and the second problem are addressed by the scale hierarchy proposed by *Stevens* [Ste46]. It distinguishes nominal, ordinal, interval, and ratio scales. *Stevens* assumes that a scale maps an empirical relation to the real numbers and discusses how the scale values can be interpreted.

- *Nominal:* The values of this scale can only be interpreted as unique identifiers. Consider, for example, two business process models that have the unique IDs 1 and 7. The only conclusion that can be drawn from this data is that the models are not the same. Accordingly, any transformation of the scale values can be performed that does not affect the uniqueness of the identifiers.

- *Ordinal:* The ordinal scale preserves an order relation that exists in the domain of the empirical relation. For a questionnaire asking in how far several business process models are complex, the responses could be mapped to the scale values 1 for trivial, 2 for rather simple, 3 for rather complex, and 4 for incomprehensible. This representation of the responses is as good as any other that does not change the order of the values. Therefore, any monotone transformation like taking logs or square roots can be applied.

- *Interval:* The interval scale is invariant to linear transformations that preserve relative distance. Consider two business process models and the point in time when they were designed. We could encode the values as days after the start of the modeling project (say, 01 September 2000) or as years AD in the range of real numbers. We could then use a linear transformation to calculate from one scale to the other.

- *Ratio:* In contrast to the interval scale, the ratio scale has a zero element that represents absence of a certain property. Consider the number of connectors a certain business process model has. Accordingly, the values can be multiplied by a constant as this preserves the relative ratio of the values.

There is a hierarchy between the scale types, since the classes of meaningful transformations narrow from nominal to ratio scale. Furthermore, *Stevens* recommends using certain statistics only for specific scale types, e.g. the mean only for interval scale data [Ste51]. Criticism against such a clerical restriction and the scale type hierarchy itself is presented in [VW93]. In [MT77] an alternative list of categories including names, ordered grades, ranks, counted fractions, counts, amounts, and balances is proposed. Beyond that, *Guttman* argues that instead of restricting data analysis to permissive statistics for a scale, one should rather consider the loss function to be minimized [Gut77]. This argument points to the problem that measurement involves validity and reliability issues.

In essence, *validity* refers to the question whether conclusions based on a measure are actually accurate, and whether measurement captures what is intended to be measured. Since abstract concepts have to be translated into a measurable operational definition or a metric, there is plenty of room for mismatch (see [Kan02, LB06]). Validity can be judged in terms of freedom from systematic error. While the true scores are often not available, validity is, in general, difficult to assess. Establishing the validity of a measurement involves the three issues of content validity, criterion validity, and construct validity (see e.g.[SA05, MDF05]). If a measurement is not valid, it is also not possible to draw valid conclusions from it.

- *Content Validity:* This type of validity refers to the ability of a measurement scale to represent the full range of meanings associated with the empirical phenomenon.

Assuming we want to find out which one of two process models is perceived to be more complex: If a questionnaire only were to offer two answers like "model A seems more complex" and "model B seems more complex", we would have a problem with content validity since the phenomenon of indifference ("both models seem equally complex") could not be represented.

- *Criterion Validity:* This type of validity points to the pragmatic value of a measurement, i.e. how closely measurement values and real phenomena are connected. Assume we are interested again in the complexity of a process model as it is perceived by modelers. If we choose to consider the number of arcs as a measurement for it, we might encounter the problem that models with the same number of arcs, for example, a sequential model and a model with arbitrary cycles, are perceived as having totally different complexity. Therefore, the number of arcs might have a problem with criterion validity related to complexity, since the measurement and the real phenomenon might not be closely connected.

- *Construct Validity:* This type of validity refers to theoretical considerations related to the question why a certain construct is appropriate. Consider again the perceived complexity of process models, and assume that the randomly assigned ID number of the model exhibits the capability to rank a sample of process models with respect to complexity. Though we might have a certain degree of criterion validity due to the pragmatic ranking capability, we have a problem with construct validity since there is no theoretical explanation available for the connection between ID and perceived complexity.

An alternative classification of measurement validity is proposed in [TD06b].

Reliability refers to the consistency of measurements over entities and time. While a scale can be reliable and not valid, an unreliable measurement cannot be valid. Therefore, reliability establishes an upper bound on validity (see e.g. [SA05, p.364]). In the following sections, we will discuss metrics that can be calculated from graphs, software artifacts, and business process models. Accordingly, reliability basically relates to the question whether our calculation algorithm works correctly and deterministically. Furthermore, there are hardly content validity problems since all metrics are based on counts

related to the models. Still, we will discuss construct validity from a theoretical point of view for each of the metrics that we identify. Finally, we will test criterion validity in the following chapter. But first we present results from the neighboring disciplines network analysis and software measurement, respectively.

5.2 Metrics in Network Analysis

Network analysis refers to structure theory and related methods in the area of applied graph theory [BE05b, p.1]. It has a long tradition in social sciences as social network analysis (SNA) dating back to the 1930s (cf. e.g. [Sco00, pp.7]), but there are also applications in engineering related to electrical circuits, or in natural sciences like epidemiology. Since EPC business process models are a special class of graphs, it is worth investigating whether network analysis techniques might be applicable.

Network analysis deals with three subareas with a focus on either elements, groups, or the overall network [BE05b]. Firstly, *element-level* analysis deals with quantifying the position of an element within a network. The Google PageRank is an example that assigns an individual web page a rank based on its connections within a network of web pages. Secondly, *group-level* analysis refers to sets of elements of a network and their relationships. An application example is the case of describing political decision making with concurrent groups of interest. Thirdly, *network-level* analysis considers properties of the overall network. Some properties have been studied related to the movie database imdb about what the average distance between any two actors is. For business process models, we are particularly interested in network-level analysis metrics. In this section we focus on degree, density, distance, centrality, and connectivity metrics. For further details on network analysis refer to [Sco00, BE05c].

A graph $G = (V, E)$ defined by a set of vertices V and edges $E \subseteq V \times V$ is the starting point of network analysis. EPCs as defined in Definition 3.11 are a specific kind of graph with the sets of functions, events, and connectors being the vertices (or nodes) and directed edges called arcs. A basic set of network analysis techniques is related to the degree of the vertices of a graph. The *degree d(v)* of a vertex is the number of edges that

are connected to it. For a directed graph the degree of a vertex is the sum of its in-degree (i.e. the number of incoming arcs) and its out-degree (i.e. the number of outgoing arcs). The average degree

$$\overline{d}(G) = \frac{1}{|V|} \sum_{v \in V} d(v)$$

summarizes whether vertices are connected to many or to few other vertices. Since the sum of degrees equals twice the number of arcs, the average degree can also be calculated as

$$\overline{d}(G) = \frac{2 \cdot |E|}{|V|}$$

A frequently used metric in network analysis is *density* (cf. [Sco00, pp.69]). Density gives the ratio of edges divided by the maximum number of possible edges. The density Δ of a directed graph is calculated as

$$\Delta(G) = \frac{|E|}{|V| \cdot (|V| - 1)}$$

ranging from 0 for an empty graph to 1 for a complete graph (assuming no self-edges from a vertex to itself). Despite its popularity the results of this metric have to be handled with care. This is in particular due to the fact that complete graphs are rare in practice [Sco00, p.70]. For a random graph, it can be expected that the degrees of the vertices are binomially distributed, i.e. only every other potential edge would be included yielding a Δ of 0.5. Several researchers observed that natural graphs do not follow such a binomial, but rather a distribution that can be expressed as a *power law* (see [BS05] for an overview). This power law defines the distribution over degrees using two parameters $c > 0$ and $\gamma > 0$ such that the expected amount of vertices with degree k is $ck^{-\gamma}$. The constant c is basically used for scaling in such a way that the sum over k yields either 1 or the number of vertices. An implication of such a power law distribution is that nodes with a small degree are much more likely that nodes with a high degree. For some example graphs, γ values are estimated in [BA99] as for the actor collaboration graph ($\gamma \approx 2.3$) and the power grid of the United States of America ($\gamma \approx 4$). The implication of this finding is twofold. Firstly, if such a power law is present, then graphs become less dense with increasing size. Secondly, and due to this fact, density might be only applicable for

a comparison of graphs of the same size. For business process models, it can be expected that nodes with a high degree value are scarce. Therefore, the implications of the power law distribution must be kept in mind when interpreting the density metric of process models.

While density captures the overall cohesion of a graph, *centrality* metrics describe to which extent the cohesion is organized around a central point [Sco00, p.89]. Several centrality metrics are based on the distance matrix $\delta(u, v)$ of a graph, in which each entry gives the shortest path between vertices u and v, also called geodesic distance. If the graph does not include cycles with negative weight, the distance matrix can be calculated in polynomial time (see [BS05] for an overview of algorithms). Based on the distances, it is easy to determine the average distance

$$\overline{\delta}(G) = \frac{1}{|V| \cdot (|V| - 1)} \sum_{\substack{u,v \in V \\ u \neq v}} \delta(u, v)$$

and the diameter

$$diam(G) = max\{\delta(u, v) \mid u, v \in V\}$$

of a connected graph[1]. The centrality of a graph depends upon the sum of differences between the score of the most central point and any other point. It is the ratio between this sum of differences and the maximum possible sum of differences (see [Fre79]). Centrality scores can be calculated based on degree sum, closeness, and betweenness. While the *sum of degrees* of a vertex only uses information of its local importance in a graph, *closeness* considers the sum of distances between vertices. The closeness centrality of a graph is defined as the inverse of the average distance that a vertex has to all other vertices. Given the maximum sum of these differences, one can calculate the closeness centrality for the overall graph (see [Fre79]). *Betweenness* is based on the idea that certain vertices are more important since they connect subparts of the graph. Accordingly, the betweenness proportion of a vertex y for a pair x and z describes the proportion of shortest paths that connect x and z via y. The pair dependency of x and y is the sum of

[1]For a disconnected graph, the calculation has to be restricted to those pairs that do not have an infinite distance.

betweenness proportions of y that involves all pairs containing x. The betweenness score of a vertex is, therefore, half the sum of all dependency scores of that vertex. The most central point of a graph can be identified using either degree sums, closeness, or betweenness scores. For further details and formulas, see [Fre79, KLTPZ05]. While the diameter appears to be interesting for measuring the length of an EPC process model the centrality concept is rather difficult to relate to business process modeling concepts. Therefore, we will disregard centrality metrics for measuring complexity of an EPC.

Connectivity is related to questions about how many vertices have to be removed from the graph to make it unconnected. An interesting concept in this context is that of a cut-vertex. A cut-vertex (also called articulation point or separation vertex) is a vertex which, when removed, increases the number of connected components of a graph. The *number of cut-vertices* can be calculated in $O(|V| + |E|)$ based on a depth-first search tree of the graph (see [CLRS01, Ch.22] or [BE05a]). It gives important information about how easily the overall graph can be split into its components. For an EPC, the number of cut-vertices might reveal information in how far the model could be understandable in a divide-and-conquer way, i.e. considering components that are connected via cut-vertices in isolation. Beyond that, depth-first search trees can also be used to decide whether a graph is planar, i.e. if it can be drawn in such a way that edges do not cross [HT74]. Since process models are frequently drawn as planar graphs, we assume that the planarity property has little selective power and, therefore, disregard it for process model.

In summary, degree, density, distance, and connectivity metrics seem to be interesting not only for graphs in general, but also for business process models in particular. Therefore, we will adapt these metrics to EPCs in Section 5.5. For further network and graph analysis techniques we refer to [Sco00, CLRS01, BE05c].

5.3 Metrics in the Software Engineering Process

In the software engineering process, several metrics are used to provide input for quality assurance (see e.g. [Boe79, Som01]). In this context, the challenge is to establish a relationship between metrics that are easy to measure, i.e. internal product attributes

which can be directly calculated for software artifacts, and both external attributes like maintainability or reliability as well as unknown parameters like total effort and number of defects. Instead of the term error, software engineering used the generic term defect comprising faults and failures (see [IEE83, FP97]). Faults are defects that result from human errors which are encoded in the software. The terms fault and error are often used synonymously. Failures are departures of a software system from its required behavior. Potentially, several failures might be tracked back to the same fault as a source. Software measurement approaches typically follow a top down design. The classical Goal-Question-Metric (GQM) approach by *Basili et al.* [BW84, BR88] advocates a clear definition of the overall goal for a software design project. Based on this goal, several questions can be derived related to how the goal can be achieved. Respective metrics offer a quantitative answer to the questions. A similar approach is proposed by *Kan* [Kan02], who suggests first to identify a concept of interest and to define it. In a second step, an operational definition has to be found that serves as the basis for a measurement related to the concept. This measurement must be checked for validity and reliability with respect to the concept. In the remainder of this section, we will present several metrics for internal attributes related to the structure of a program, like size and complexity, which are frequently used in software engineering. These metrics essentially capture control flow, data flow, and/or data structure. After that, we will discuss external quality attributes. External quality attributes model aspects like reliability, usability, and maintainability and relate them to internal attributes. Furthermore, we report on empirical results related to validating these metrics.

Before presenting individual metrics, we must discuss the association between a software program and its control flow graph. Since we aim to investigate the potential analogy between software measurement and business process model measurement we consider software programs to be represented as control flow graphs. A control flow graph $G = (V, E)$ can be derived from a program by mapping blocks of sequential code to vertices with one input and one output arc, and branching statements like `if` or `while` to vertices with multiple input and output arcs (cf. [MB89]). Metrics defined for control flow graphs can be easily adapted to process models.

The *lines of code* metric is the traditional metric for measuring the size of software.

While the idea of counting the lines of a software program is rather simple, there have been several discussions whether, for example, comments should be included. *Jones* states that depending on the way how lines are counted, the results might differ by the factor of five [Jon86]. Standardization efforts, such as the 242 pages long report of the Software Engineering Institute of Carnegie Mellon University, illustrate the extent of choices in this area [SEI92]. Given the control flow graph $G = (V, E)$ of a program, the lines of code can be identified with the number of vertices:

$$LoC(G) = |V|$$

One problem with lines of code is that code from different programming languages is not directly comparable. Research on language productivity has established so-called gearing factors that capture the effort of writing an arbitrary program in a certain language, for example, programming in SQL requires an average effort of 39 compared to 172 in Assembler [LB06, p.38]. Beyond that, an increase in lines of code does not necessarily imply an increase in functionality. Function Point Analysis originally proposed in [Alb79] addresses this problem by assigning a score to each input, output, interface, data file, and inquiries based on their individual difficulty. While function points can be compared for programs written in different languages, there is the problem that function point analysis requires human interpretation of program difficulty. Therefore, it can only be partially automated. This is a major disadvantage compared to lines of code that can be counted automatically.

The *cyclomatic number* CC proposed in [McC76, MB89, MW94] is an early complexity metric. It is based on the control flow graph $G = (V, E)$ of a program and captures the number of paths through a program that are needed to visit all vertices. More precisely, it matches the maximum number of linearly independent paths through a program. The cyclomatic number is of particular importance for test theory since it defines the number of test cases required for unit tests. It can either be calculated based on the number of vertices and edges as

$$CC(G) = |E| - |V| + 1$$

or alternatively by counting the number of choices weighted with the number of alternatives. Since CC is not biased towards a specific programming language, it can be used to compare the complexity of programs written in different languages. The cyclomatic complexity density CCD is an CC extension for predicting maintenance productivity [GK91]. CCD is calculated as CC divided by the lines of code LoC. A second extension called essential cyclomatic complexity ECC is a measure of unstructuredness of the program [McC76]. It is calculated as the cyclomatic number of the code after removing all structured constructs like `if` and `while` statements. A totally structured program, therefore, has an essential cyclomatic complexity of zero. The cyclomatic number can be calculated for EPC business process models that do not include concurrency (see Figure 5.1).

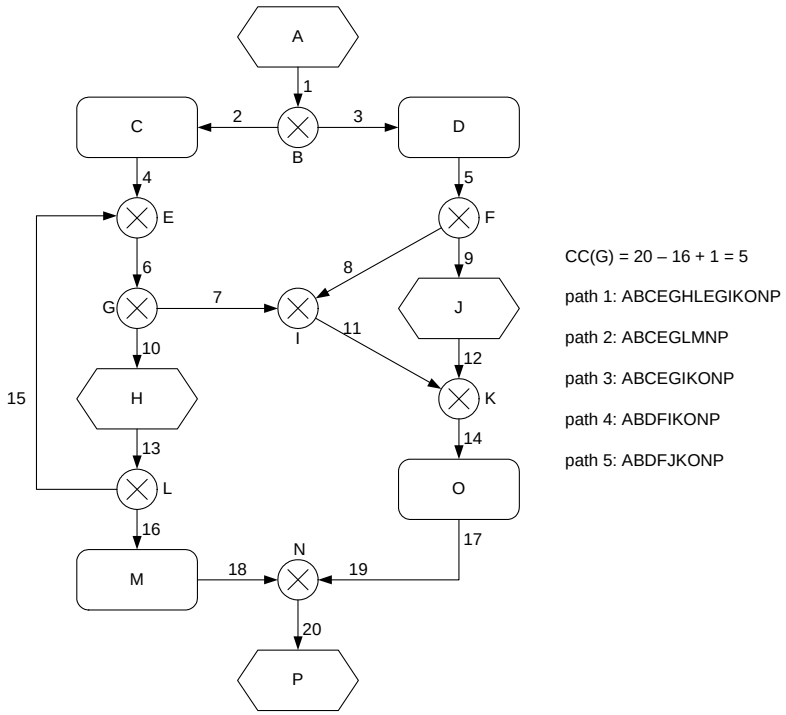


Figure 5.1: Cyclomatic number of an EPC without concurrency

Halstead's metrics provide measurable definitions for the length, vocabulary, volume, difficulty, and effort of a software program based on the number of operators and operands [Hal77]. Operators comprise commands and structuring elements like parenthe-

ses. Operands basically refer to elements that have a value like variables and constants. The operator parameters n_1 and N_1 refer to the number of distinct operators and the total occurences of operators, respectively. The operand parameters n_2 and N_2 describe the number of distinct operands and the total occurences of operands, respectively. *Halstead* then defines the following metrics:
Length $N = N_1 + N_2$
Vocabulary $n = n_1 + n_2$
Volume $V = N \cdot log_2(n)$
Difficulty $D = n_1/2 \cdot N_2/n_2$
Effort $E = D \cdot V$
The formula of volume is motivated by the number of binary mental comparisons needed to write a program of length N. The formula for difficulty is basically a postulate of *Halstead*'s theory. Furthermore, *Halstead* claims with reference to the psychologist *Stroud* that the required programming time T can be calculated as $T = E/18sec.$ Although the work by *Halstead* had a lasting impact on software metrics research, it has been criticized repeatedly as "an example of confused and inadequate measurement" [FP97].

The *information flow metric* proposed by *Henry and Kafura* [HK81] is based on the data flow between different modules of a program. Information can be passed between modules in three ways: firstly, by a module invoking another one; secondly, by returning a result from the invoked module to the caller; and thirdly, by exchanging information via a global data structure. The *fan-in* of a module M refers to the number of calls to the module plus the number of reads to global data structures. The *fan-out* captures calls to other modules and write operations to global data structures. Based on these parameters, the information flow complexity is calculated as

$$IFC(M) = LoC(M) \cdot (fan_{in} \cdot fan_{out})^2$$

The IFC metric can be used at design-time to predict which modules are likely to contain errors. The multiplication of fan-in and fan-out has been criticized in [FP97]. If a module has either no fan-in or fan-out the metric yields zero which is misleading.

The lines of code, cyclomatic complexity, *Halstead* complexity, and information flow

complexity metric were developed for procedural programming languages. In [CK94], a set of metrics for object-oriented development is proposed. In this context, coupling and cohesion are important concepts. Coupling describes the degree of interdependence between a pair of modules, and cohesion describes the degree to which the elements of a module are functional related (see e.g. [YC79, Som01]). The proposal is based on ontological work of *Bunge*, *Wand*, and *Weber* [Bun77, WW90] and includes six metrics.

- *Weighted methods per class:* This metric is calculated as the sum of complexity weights over all methods.
- *Depth of inheritance tree:* The inheritance tree defines a hierarchy of classes from which an individual class inherits properties and methods. The metric gives the depth of such inheritance for a class in order to describe how many ancestors potentially affect it.
- *Number of children:* This metric gives the number of successors in the inheritance tree for a particular class.
- *Coupling between object classes:* This measure states to how many other classes a class is coupled.
- *Response for class:* This metric gives the size of the set of methods that a class might use to respond to a message. It is calculated as the sum of local methods plus methods called by these local methods.
- *Lack of cohesion metric:* This is the number of disjoint sets identified by comparing each pair of methods and the instance variables they relate to. A high value for this metric would suggest to split up the class.

A generic framework for classifying further object-oriented metrics is proposed and validated against 350 metrics in [VPL07].

The previously presented metrics related to internal attributes of a software artifact were tested in various empirical studies for their criterion validity with respect to predicting defects or project effort. *Fenton and Ohlsson* test several hypotheses on faults

and failures, among others that LoC is a useful predictor for defects and that complexity metrics perform better [FO00]. They find that LoC is indeed suited to rank modules according to their absolute number of faults. Yet, they also confirm the finding of *Basili and Perricone* [BP84] that fault density seems to decrease with module size. Furthermore, cyclomatic complexity CC does not seem to outperform LoC as a predictor for errors. *Fenton and Ohlsson* propose to rather consider CC as a predictor for comprehensibility and maintainability.

An overview of effort estimation is given in [LB06, pp.79]. Benchmark data, in particular, is useful to assess productivity, for example, in one staff month code produced for data processing applications appear to range between 165 and 500 lines [LB06, p.86]. Measurements are also used to predict the dynamics of effort and defects throughout the development process. Given a record of found defects and a typical distribution of defects over time, one can estimate defects to be found in future. A typical distribution in this context is the Rayleigh curve that sharply steepens to a maximum and then decreasing falls towards zero. Such an estimation is particularly helpful for determining a suitable release date (cf. e.g. [LB06]). Regression-based prediction can also be applied with respect to effort. *Boehm's* constructive cost model (COCOMO) assumes that effort $E = aS^bF$ with S measuring size, F being an adjustment factor, and a and b depend on the type of software being constructed [Boe81]. The ISO/IEC 9126 Software Product Quality Model [ISO91] provides an overall classification scheme for quality-related product attributes on four levels. It can be used as a framework for several metrics defined earlier. Further models and empirical results are summarized in various text books on software measurement, as for example [Zus91, FP97, Kan02, LB06].

5.4 Related Work on Metrics for Process Models

In this section, we discuss related work on metrics for process models. Since the authors hardly refer to each other, it is difficult to group the contributions according to subject area, so their work will be presented chronologically.

In 1990, *Lee and Yoon* conducted pioneering work on the definition of metrics for

Petri net process models and their empirical validation [LY90, LY92]. The authors group their metrics into two categories: *structural metrics* include simple counts of places, transitions, arcs and the cyclomatic number of the control graph. *Dynamic metrics* cover the number of markings and the maximum and average number of tokens for the original and a reduced state space. In an empirical study of a set of 75 Petri nets, the authors find that a simple adaptation of the cyclomatic number is not a suitable complexity metric; that the number of places can serve as a predictor for the size of the reachability graph in an exponential equation; and that reduction techniques reduce the analysis problem by factor four for the sample nets.

Nissen was among the first to introduce measurement concepts for business process modeling and business process design in particular (see [Nis94, Nis96, Nis98]). His work is motivated by the observation that business process design is an ill-structured process, and that it is therefore a case for business process reengineering on itself. Based on a set of measurements for process models, he utilizes design heuristics and a knowledge-based system for reasoning on the quality of the design. The proposed metrics cover counts for *distinct paths*, *hierarchy levels*, *nodes* in the process model, *cycles*, *diameter*, and *parallelism* as number of nodes divided by the diameter. These metrics are provided by the KOPeR tool, which guides the process reengineering process [Nis98].

Tjaden, Narasmihan, and Gupta operationalize four characteristics of a business process that need to be balanced: simplicity, flexibility, integration, and efficiency [TNM96, Tja01]. *Simplicity* is calculated based on so-called basic complexity of the process, i.e. in essence the sum of nodes, arcs, and roles. The overall simplicity is then formalized as the average activity complexity divided by the maximum activity complexity. *Flexibility* and *Integration* are determined based on a list of scores similar to function point analysis. Though it is mentioned, the technical report [Tja01] does not operationalize efficiency. *Balasubramanian and Gupta* criticize the high level of abstraction of the metrics by Tjaden (see [BG05]). Furthermore, a function point approach, such as it is proposed for flexibility and integration, is difficult to automate.

Building on measurement efforts for concurrent programs (see e.g. [Sha88]), *Morasca* proposes a set of metrics for Petri nets and a theoretical foundation [Mor99]. He identifies size, length, structural complexity, and coupling as interesting attributes

of a Petri net, and for each attribute, he defines a set of axiomatic properties which a respective metric would have to fulfill. For size he proposes *number of places and transitions*, for length the maximum minimal distance (i.e. the *diameter*), for structural complexity the *number of base paths*, the *concurrent contribution* as the number of arcs minus two times the number of transitions, the *sequential contribution* as the number of transitions with one input places, and for coupling the *number of inbound or outbound arcs of a subnet*. Since *Morasca's* contribution is theoretical, an empirical validation is not included in his paper.

Latva-Koivisto [LK01] proposes several complexity metrics for business process models including the *Coefficient of Network Connectivity*, *Cyclomatic Number*, *Reduction Complexity Index*, *Restrictiveness Estimator*, and *Number of Trees in a Graph* as metrics. Basically, this work has two weaknesses. Firstly, there is no distinction made between different kinds of routing elements. Secondly, a motivation to consider the restrictiveness estimator and the number of trees in a graph as a complexity metric is missing. The work by *Latva-Koivisto* did not receive too much attention in the community since it was published only as a technical report.

In [Rei03, RV04] *Reijers and Vanderfeesten* develop a set of coupling and cohesion metrics to guide the design of workflow processes. The coupling of an activity is calculated based on its relation cohesion and its information cohesion. The *activity relation cohesion* λ describes in how far an activity shares control flow input and output with other activities. It is defined as the sum of overlaps between each activity pair divided by the maximum number of pairs with choices not being considered as overlap. The *activity information cohesion* μ determines how many information objects are used by an activity more than once in relation to all information objects in use. The cohesion of an activity is calculated as $\lambda \cdot \mu$. The overall *process cohesion* c is then defined as the average activity cohesion. The *process coupling* k metric is calculated as the number of connected activities divided by the maximum number of connections, i.e. process coupling is basically equated with the density Δ of the process graph. Inspired by [SB91] the authors define a process coupling/cohesion ratio as $\rho = k/c$ with a low value indicating good design.

The work of *Cardoso* is centered around an adaptation of the cyclomatic number for business processes he calls *control-flow complexity (CFC)* [Car05a, Car05d]. CFC is

calculated from a model by summing up the split connectors weighted by the combinations of output markings they can produce, i.e. 1 for an AND-split S_{and}, $|S_{xor} \bullet|$ for an XOR-split S_{xor}, and $2^{|S_{or}\bullet|} - 1$ for an OR-split. The CFC metric was validated against a set of complexity axioms proposed by *Weyuker* [Wey88] in [Car05b, Car05c], and tested with respect to their correlation with perceived complexity [Car06]. Yet, in an analysis of the SAP Reference Model, the CFC metric was found unsuitable for the prediction of errors in business process models [MMN+06c].

The research conducted by a group including *Canfora, Rolón, García, Piattini, Ruiz, and Visaggio* extends work related to measurement of the software process. In [CGP+05] *Canfora et al.* present a set of metrics and evaluate their suitability to serve as predictors for maintainability of the process model. The operational definition of maintainability covers *analyzability* as the likeliness of discovering errors or deficiencies in the model, *understandability* as the likeliness of comprehending a model, and *modifiability* as the likeliness of correctly changing the structure of the model. The metrics address structural properties of the process model related to size, complexity, and coupling.

- *Size:* count metrics for the number of activities, work products, process roles, input and output dependencies of work products with activities, ratio of work products to activities, and ratio of process roles to activities.

- *Complexity:* Number of dependencies between work products and activities, and ratio between input or output dependencies of work products with activities and total number of dependencies.

- *Coupling:* Number of precedence dependencies between activities and activity coupling.

In a set of related experiments, the authors find that most of the metrics are correlated to maintainability: The numbers of activities, work products, work product dependencies, and precedence dependencies show good results while activities coupling and the ratio of roles to activities are only correlated to a limited extent. The number of roles and the ratios related to work product dependencies are not correlated. *Rolón et al.* extend this set

of metrics and tailor it to the specifics of BPMN reflecting additional numbers for events, gateways, message flows, and pools [ARGP06a, ARGP06b, ARGP06c, AGRP07].

Inspired by the works of *Nissen* and *Tjaden*, *Balasubramanian and Gupta* propose a set of metrics to support business process design (see [BG05]). This set includes, among others, metrics to quantify the degree of automatic decision making (*branching automation*), *activity automation*, *role integration*, *role dependency*, *activity parallelism*, and *transition delay risk*. The metrics are applied in a case study for identifying the best of two alternative process designs.

Only partially related is the work by *Etien and Rolland* on measuring the fitness of a business process model with a real-world process [ER05]. The authors define several metrics related to concepts of the Bunge-Wand-Weber ontology, which express in how far certain concepts are represented in a model that is present in the real world. In practice, such a measurement maps to a comparison of a business process model with a potentially more complete model that captures entities of the BWW-ontology related to the real-world process. Accordingly, such a measurement is more of theoretical interest than of benefit in the design process.

The two survey papers by *Cardoso, Mendling, Neumann, and Reijers* [CMNR06] and *Gruhn and Laue* [LG06] summarize a great share of the earlier mentioned work. They reveal that several metrics for business process models are basically adaptations of software complexity metrics. This also holds true for an adaptation of complexity metrics for EPCs as reported in [GLM06]. Only some authors provide operationalizations for central concepts of business process models such as parallelism, cycles, and sequentiality (see e.g. [Nis98]). Several process-related aspects like structuredness and mismatch of connectors are covered only to a limited extent, or not at all. In the following section, we will identify several metrics that are specifically tailored for predicting errors in business process models.

5.5 Definition of Error Metrics for Process Models

In the previous Sections 5.2–5.4, we have discussed several metrics for business process models. Many of them aim to operationalize the structural complexity of the process graph. The term complexity is discussed in software measurement and authors try to identify axioms that a complexity metric would have to fulfill (see the work by *Weyuker* [Wey88]). Such an axiomatic approach is criticized from different perspectives. In [CS91] the authors show that the *Weyuker* axioms do not guarantee that the metric is meaningful by defining an obviously useless metric that meets the axioms. More serious is the criticism by *Zuse* who shows that the aspects subsumed under the term complexity cannot be captured by a single metric alone [Zus91]. *Fenton and Pfleeger* reinforce this finding by pointing to measurement theory [FP97, pp.322], saying that every valid measurement should obey the rules of representational theory of measurement. This implies that first, an empirical concept or relationship in the real-world should be identified, followed by suitable measurements. in other words, the concept guides the measurement, and the metric does not define the concept. This principle is incorporated in several software measurement approaches, such as the Goal Question Metric (GQM) approach by *Basili et al.* [BW84, BR88] or the Concept, Definition, Operational Definition, Measurement approach by *Kan* [Kan02] that we mentioned before.

Following this line of argumentation, we consider the comprehensibility of the business process model as the main determinant for error probability. This is motivated by the assumption that process models are constructed by human modelers and that their design is subject to bounded rationality [Sim96]. The comprehensibility of any model by a person is influenced by a variety of factors including *model-related* factors like size, *personal factors* like modeling competence, *purpose* of modeling like documentation or execution, *domain knowledge*, *modeling language*, or *graphical layout* of the model. In this chapter we only investigate the model-related aspect. More precisely, we analyze several metrics that capture various aspects related to either the *process model structure*, the *process model state space*, or both (cf. [LY92]) and discuss their impact on error probability. Each metric is presented by giving its (1) symbol, (2) definition, (3) rationale why it should be considered, (4) limitations, (5) the hypothesis related to it, and (6)

related work that mentions similar metrics in previous research. In the following, we consider a business process model to be a special kind of graph $G = (N, A)$ with at least three node types $N = T \cup S \cup J$, i.e., *tasks* T, *splits* S, and *joins* J, and *control flow arcs* $A \subseteq N \times N$ to connect them. We use the generic term connectors $C = S \cup J$ for splits and joins collectively. Each connector has a *label* AND, OR, or XOR that gives its routing of merging semantics. For presentation purposes, we subdivide the set of metrics into the categories *size*, *density*, *partitionability*, *connector interplay*, *cyclicity*, and *concurrency*.

5.5.1 Size

Several papers point to *size* as an important factor for the comprehensibility of software and process models (see e.g. [Nüt95, CGP+05, CMNR06, LG06, Zus91, FP97, Kan02]). While the size of software is frequently equated with lines of code, the size of a process model is often related to the number of nodes N of the process model. Furthermore, we consider the *diameter* of the process graph.

Symbol	S_N
Definition	Number of nodes of the process model graph G
Metric	$S_N(G) = \|N\|$
Rationale	A larger business process model in terms of $S_N(G)$ should be more likely to contain errors than a small one, since the modeler would only be able to perceive a certain amount of nodes in a certain period of time.
Limitations	There are obviously large models in terms of $S_N(G)$ that are unlikely to have errors, e.g. if the model is sequential without any connectors.
Hypothesis	An increase in $S_N(G)$ should imply an increase in error probability of the overall model.
Related Work	LoC [Zus91, FP97, Kan02]; Number of nodes [LY92, Nis98, Mor99, BG05, CGP+05, CMNR06, LG06, ARGP06c].

The size metric S_N does not differentiate between the several node types and its subsets of an EPC. Accordingly, we define size metrics for each EPC element type and its subsets by mentioning it as the index of S, i.e. S_{E_E} refers to the number of end events of an EPC, and S_F to the number of functions of an EPC. Still, the size of the model might be closer related to the longest path of it. Therefore, we define the diameter $diam$ of a process model.

Symbol	$diam$
Definition	The diameter gives the length of the longest path from a start node to an end node in the process model.
Metric	$diam(G)$ is calculated either based on the distance matrix of a process model or based on shortest path algorithms (see [BS05])
Rationale	A larger business process model in terms of $diam(G)$ should be more likely to contain errors than a small one, since the modeler would only be able to perceive a certain amount of consecutive nodes in a certain period of time.
Limitations	Obviously, there are large models in terms of $diam(G)$ that are unlikely to have errors, e.g. if the model is sequential.
Hypothesis	An increase in $diam(G)$ should imply an increase in error probability of the overall model.
Related Work	The diameter was also proposed in [Nis98, Mor99].

Figure 5.2 illustrates one particular shortcoming of size as a simple count metric. The left and the right model have exactly the same number of nodes, but the diameter of the right EPC is longer. This difference between the models does not become apparent if only the size metric S_N is considered.

5.5.2 Density

In the context of this thesis, we use density as a generic term to refer to any metric that relates numbers of nodes to numbers of arcs. There are several metrics that provide

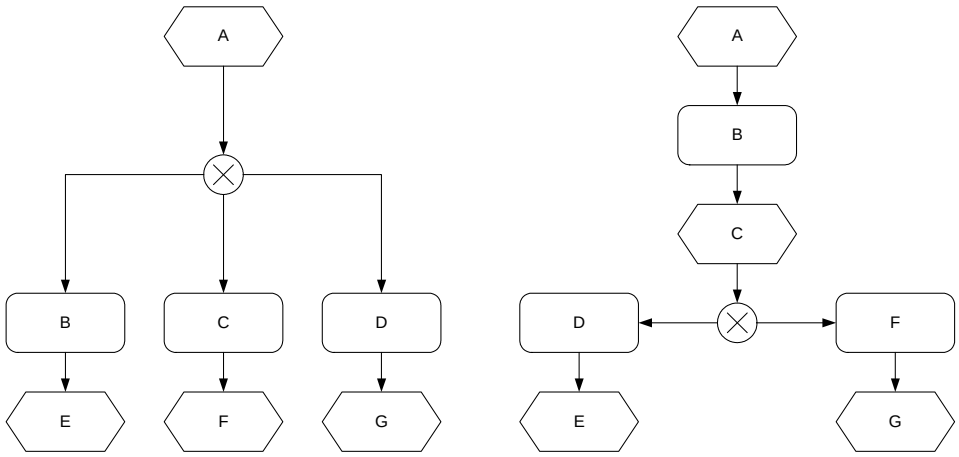


Figure 5.2: Two EPCs of the same size with different diameter

information about the relation of arcs and nodes. Here, we consider the *density* metric, the *coefficient of network connectivity*, *average connector degree*, and *maximum degree*.

Symbol	Δ
Definition	The density of the process graph refers to the number of arcs divided by the number of the maximum number of arcs for the same number of nodes.
Metric	$\Delta(G) = \frac{\lvert A\rvert}{\lvert N\rvert \cdot (\lvert N\rvert - 1)}$
Rationale	A business process model with a high density $\Delta(G)$ should be more likely to contain errors than a less dense model with the same number of nodes.
Limitations	The density in terms of Δ is difficult to compare for models with a different number of nodes: larger models with the same average degree have a smaller density since the maximum possible number of arcs grows by the square of $\lvert N\rvert$. Furthermore, many natural graphs seem to obey a power law (see [BE05c]) which implies that larger models would be less dense in terms of Δ.
Hypothesis	An increase in $\Delta(G)$ should imply an increase in error probability of the overall model.

Related Work	Δ is mentioned as process coupling metric in [RV04].

Symbol	CNC
Definition	The coefficient of connectivity gives the ratio of arcs to nodes.
Metric	$CNC(G) = \frac{\|A\|}{\|N\|}$
Rationale	A denser business process model in terms of $CNC(G)$ should be more likely to contain errors since the modeler has to perceive more connections between nodes than in a model that is less dense.
Limitations	There are process models with the same CNC value that might differ in error probability. Consider e.g. a sequential model without any connector and a model with the same number of nodes having one split-connector. The two models also have the same number of arcs, therefore CNC is equivalent.
Hypothesis	An increase in $CNC(G)$ should imply an increase in error probability of the overall model.
Related Work	CNC is listed in [LK01, CMNR06]. The inverse of CNC called activity coupling $NCA = \|N\|/\|A\| = 1/CNC$ is proposed in [CGP+05, ARGP06c].

Symbol	$\overline{d_C}$
Definition	The average degree of connectors gives the number of nodes a connector is in average connected to.
Metric	$\overline{d_C}(G) = \frac{1}{\|C\|} \sum_{c \in C} d(c)$
Rationale	A denser business process model in terms of $\overline{d_C}(G)$ should be more likely to contain errors since the modeler has to perceive more connections between nodes than in a model that is less dense.
Limitations	There are process models with the same $\overline{d_C}$ value which differ in size. Therefore, they should also differ in error probability.
Hypothesis	An increase in $\overline{d_C}(G)$ should imply an increase in error probability of the overall model.

Related Work	$\overline{d_C}$ is related to the information flow metric by *Henry and Kafura* [HK81] and its adaptation to process models in [CMNR06, LG06].

Symbol	$\widehat{d_C}$
Definition	The maximum degree of a connector.
Metric	$\widehat{d_C}(G) = max\{d(c) \mid c \in C\}$
Rationale	A business process model with a high maximum degree $\widehat{d_C}(G)$ should be more likely to contain errors since the modeler has to perceive more connections between the connector of maximum degree than in a model that has a lower maximum degree.
Limitations	There are obviously process models with a high maximum degree and the same $\widehat{d_C}$, but with a low average degree.
Hypothesis	An increase in $\widehat{d_C}(G)$ should imply an increase in error probability of the overall model.
Related Work	$\widehat{d_C}$ is closely related to the information flow metric by *Henry and Kafura* [HK81]. The idea of information flow is to identify modules whose interactions are difficult to comprehend and the connector of maximum degree is the most difficult following this line of argumentation.

Figure 5.3 illustrates the sensitivity of the different density metrics to size. The EPC on the left-hand side has 18 nodes and 17 arcs, while the one on the right-hand side only has 8 nodes and 7 arcs. The density metric Δ of the smaller model is more than twice as high (0.055 to 0.125) than that of the larger one, although the structure is quite similar. This fact reinforces the statements of Section 5.2 on the difficulty to compare graphs of different size by the density value. The CNC metric reflects the similar structure much better with similar values (0.945 to 0.875). The average and the maximum degree of connectors $\overline{d_C}$ and $\overline{d_C}$ yield 3 for both models. In this case, they nicely underline the similarity of the models.

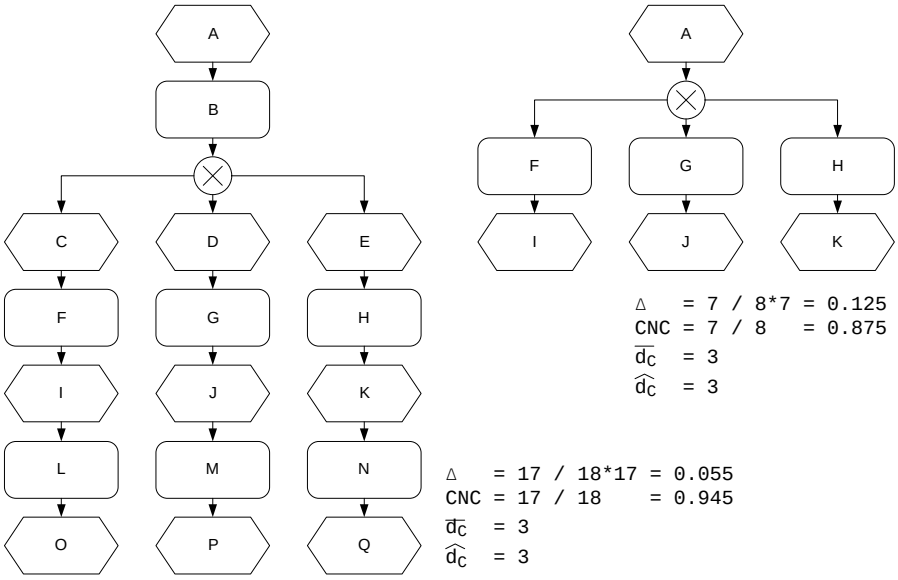


Figure 5.3: Two EPCs of the same average and maximum connector degree and varying density and CNC values

5.5.3 Partitionability

We use the term partitionability for referring to those aspects of a process model that relate to the relationship of subcomponents to the overall model. In particular, we discuss *separability* and *sequentiality*, which both capture in how far certain parts of the model can be considered in isolation. Furthermore, *structuredness* quantifies to which degree components are composed in a structured way, while *depth* defines how far a certain node is from a start or end event.

Separability is closely related to the notion of a cut-vertex (or articulation point), i.e., a node whose deletion separates the process model into multiple components. We define the separability ratio as the number of cut-vertices to number of nodes. Cut-vertices can be found using depth-first search.

Symbol	Π
Definition	The separability ratio relates the number of cut-vertices to the number of nodes.

Metric	$\Pi(G) = \frac{\lvert\{n \in N \mid n \text{ is cut-vertex}\}\rvert}{\lvert N\rvert - 2}$
Rationale	A model with a high ratio of cut-vertices should be less likely to contain errors than a model with a low ratio. If every node except the start and the end node is a cut-vertex, the model is sequential and should, thus, be easy to understand.
Limitations	The separability ratio Π can be low if there are two long sequential paths in parallel, since none of the parallel nodes is a cut-vertex.
Hypothesis	An increase in $\Pi(G)$ should imply a decrease in error probability of the overall model.
Related Work	Π has not yet been considered as a business process model metric.

Sequentiality relates to the fact that sequences of consecutive tasks are the most simple components of a process model. The sequentiality ratio relates arcs of a sequence to the total number of arcs.

Symbol	Ξ
Definition	The sequentiality ratio is the number of arcs between none-connector nodes divided by the number of arcs.
Metric	$\Xi(G) = \frac{\lvert A \cap (T \times T)\}\rvert}{\lvert A\rvert}$
Rationale	A process model with a high sequentiality ratio should be less likely to contain errors than one with a low sequentiality ratio. In contrast to the separability ratio Π, the sequentiality ratio Ξ also considers sequences that are in parallel or exclusive. If every arc connects only non-connector nodes, the model is sequential and the sequentiality ratio is 1.
Limitations	There are models with the same sequentiality ratio, but whose non-sequential arcs might differ in their degree of comprehensibility.
Hypothesis	An increase in $\Xi(G)$ should imply a decrease in error probability of the overall model.
Related Work	Ξ has not yet been considered as a business process model metric. It is related to sequential contribution of *Morasca* [Mor99].

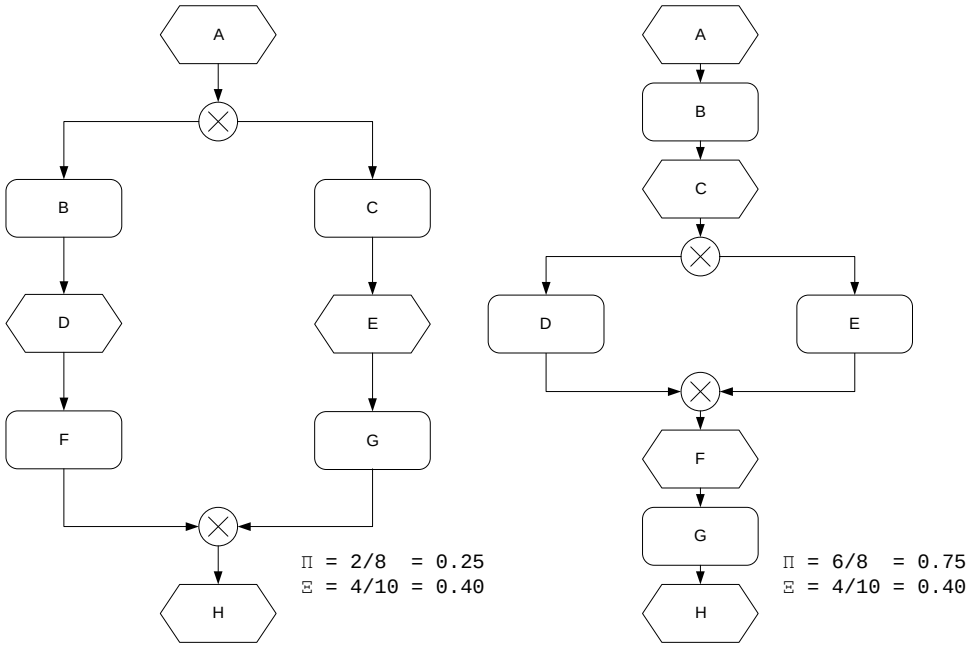


Figure 5.4: Two EPCs with the same sequentiality and different separability

Figure 5.4 illustrates the difference between the separability and the sequentiality ratio. The model on the left-hand side includes two cut vertices (the XOR-split and the XOR-join), while the model on the right-hand side additionally has the cut vertices B, C, F and G. Therefore, even though the number of nodes is the same in both models, the separability ratio is 0.25 for the left model and 0.75 for the right model. The sequentiality ratio counts sequence arcs no matter in which part of the EPC they appear. Therefore, the sequential components B to F and C to G contribute as much to the sequentiality ratio of the left model as the sequences A to C and F to H in the right model, i.e. 40%.

Structuredness relates to how far a process model can be built by nesting blocks of matching join and split connectors (see e.g. [KHB00]). The degree of structuredness can be determined by applying reduction rules and comparing the size of the reduced model to the original size. In particular, we consider the reduction of trivial constructs, structured blocks and loops, and of structured start and end components as defined in Section 4.3.2.

Symbol	Φ
Definition	The structuredness ratio of the process graph is one minus the number of nodes in the reduced process graph G' divided by the number of nodes in the original process graph G.
Metric	$\Phi_N = 1 - \frac{S_N(G')}{S_N(G)}$
Rationale	A process model with a high structuredness ratio should be more likely to contain errors than one with a low ratio. If every node is part of a structured block, the model is structured and the structuredness ratio is 1.
Limitations	There are models with the same structuredness ratio, but which differ in their degree of comprehensibility, e.g. if one is larger.
Hypothesis	An increase in $\Phi(G)$ should imply a decrease in error probability of the overall model.
Related Work	Φ has not yet been formalized in literature, but mentioned in [LG06, GL07]. It is related to essential cyclomatic complexity [McC76].

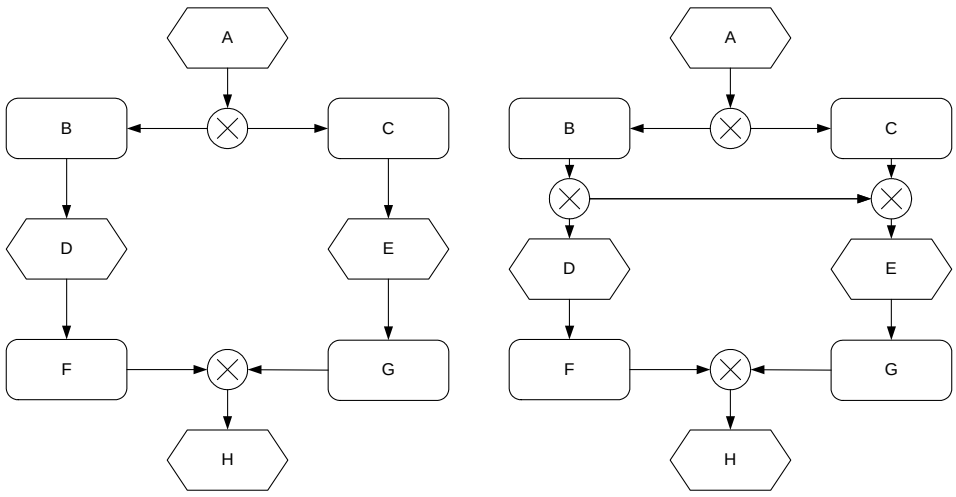


Figure 5.5: Two EPCs with the same functions and events but different degree of structuredness

Figure 5.5 illustrates the structuredness ratio by the help of two EPCs. The EPC on the left-hand side is totally structured and yields a structuredness value[2] of 1. The EPC on the right-hand side contains one additional arc and two XOR-connectors that affect the structuredness. While the nodes B to G are deleted by the reduction rule for trivial constructs, the connectors cannot be eliminated. Accordingly, only 6 out of 12 nodes are deleted, which yields a structuredness ratio of 0.5. In the EPC on the right-hand side, the arc between the left and the right column blocks the application of further reduction rules.

Depth relates to the maximum nesting of structured blocks in a process. To calculate depth also for unstructured models, we define an algorithm that calculates the in-depth $\lambda_{in}(n)$ of a node n relative to its predecessor nodes $\bullet n$. First, all nodes are initialized with an in-depth value of 0. Then, the process model is traversed along all paths starting from a start node and ending either with an end node or a node that was visited before in this path.[3] At each visited node n, the new in-depth value $\lambda'_{in}(n)$ is updated, based on the value of the previously visited predecessor node $\lambda_{in}(pre)$ and the current value $\lambda_{in}(n)$ according to the following rule:

$$\lambda'_{in}(n) = \begin{cases} max(\lambda_{in}(n), \lambda_{in}(pre) + 1) & \text{if } pre \in S \wedge n \notin J \\ max(\lambda_{in}(n), \lambda_{in}(pre)) & \text{if } pre \in S \wedge n \in J \\ max(\lambda_{in}(n), \lambda_{in}(pre)) & \text{if } pre \notin S \wedge n \notin J \\ max(\lambda_{in}(n), \lambda_{in}(pre) - 1) & \text{if } pre_i \notin S \wedge n \in J \end{cases}$$

This definition of in-depth basically captures the maximum number of split connectors that have to be visited to arrive at a node minus the number of joins on this path. The out-depth $\lambda_{out}(n)$ is defined analogously with respect to the successor nodes and decreased for splits and increased for joins. In a structured model, $\lambda_{in}(n)$ equates with $\lambda_{out}(n)$. We define depth $\lambda(n)$ as the minimum of in-depth $\lambda_{in}(n)$ and out-depth $\lambda_{out}(n)$. The depth of the process model Λ is then the maximum depth $\lambda(n)$ over n.

[2]Please note that if the reduced EPC has the minimum size of 2, i.e. only one start and one end event connected by an arc, the value of Φ is set to 1.

[3]This definition addresses potential problems with the existence of a fixed point by visiting each node only once for each path even if it is on a loop.

Symbol	Λ
Definition	The depth is the maximum depth of all nodes.
Metric	$\Lambda(G) = max\{\lambda(n) \mid n \in N\}$
Rationale	A process model with a high depth should be more likely to contain errors since connectors are deeply nested.
Limitations	There are models with the same depth that differ in size.
Hypothesis	An increase in $\Lambda(G)$ should imply an increase in error probability of the overall model.
Related Work	Λ has not yet been formalized, but mentioned in [Nis98, LG06].

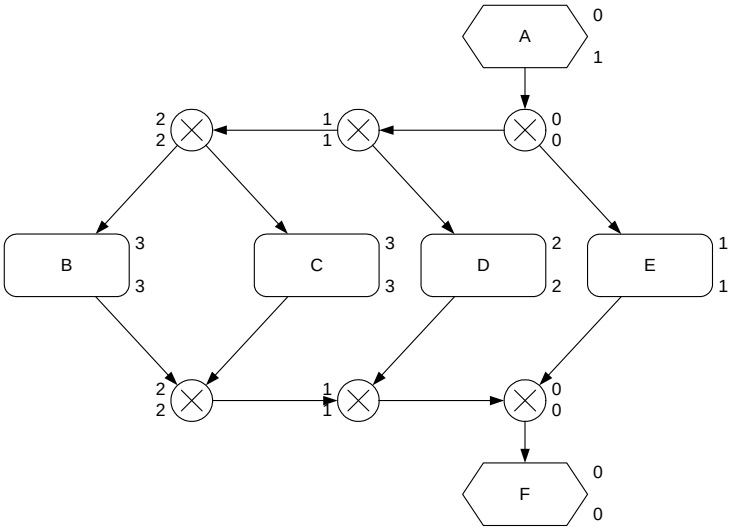


Figure 5.6: EPC with depth values next to nodes (top: in-depth, bottom: out-depth)

Figure 5.6 shows a structured EPC with its in-depth and out-depth values next to the nodes. It can be seen that both in-depth and out-depth increase with each visit to an XOR-split, while visiting an XOR-join decreases both parameters. In this context, it must be mentioned that joins are used as entries to loops and splits as exits. The depth calculation algorithm is not able to identify such loops as deeper nested structures. Still, it offers a way to quantify the depth of any process model, no matter if it is structured or not.

5.5.4 Connector Interplay

In this section, we present metrics related to connectors and their interplay. In particular, we discuss connector mismatch, connector heterogeneity, and the control flow complexity metric.

Structuredness implies that each split-connector matches a corresponding join of the same type. A mismatch might be the source of an error. Depending on the connector label and the degree, we define the connector mismatch as $MM_l = |\sum_{c \in S_l} d(c) - \sum_{c \in J_l} d(c)|$ where l is the connector type. The mismatch ratio MM gives the sum of mismatch for each connector type.

Symbol	MM
Definition	The connector mismatch gives the sum of mismatches for each connector type.
Metric	$MM(G) = MM_{or} + MM_{xor} + MM_{and}$
Rationale	A process model with a high mismatch is likely to include errors since parallel tokens might not be synchronized or alternative branches might run into AND-joins and deadlock. If the model is structured with matching split- and join connectors MM is zero.
Limitations	Languages like EPCs offer multiple start and end events. Therefore, they might show a mismatch without having errors.
Hypothesis	An increase in $MM(G)$ should imply an increase in error probability of the overall model.
Related Work	MM has not yet been formalized in literature.

Connector heterogeneity refers to which extent different connectors are used in a business process model. For defining a suitable metric that ranges from 0, for the case that there are only connectors of one type, to 1, for the case that there are the same amount of connectors of all three types, we refer to the information entropy measure which has exactly these characteristics. In contrast to the original work of Shannon and Weaver [SW63], we do not consider a binary encoding, but a ternary because of the three connector types. Therefore, the base of the logarithm is three, not two. Furthermore,

we utilize the relative frequency $p(l) = |C_l|/|C|$. Then, the connector heterogeneity is calculated in analogy to information entropy as the negative sum over the three connector types of $p(l) \cdot log_3(p(l))$.

Symbol	CH
Definition	The connector heterogeneity gives the entropy over the different connector types.
Metric	$CH(G) = -\sum_{l \in \{and,xor,or\}} p(l) \cdot log_3(p(l))$
Rationale	A process model with a high heterogeneity is likely to include errors since connectors can be mismatched more easily. If the model includes only one connector type, then CH is 0.
Limitations	Process models might have a high connector heterogeneity, but if the model is structured, an error is less likely.
Hypothesis	An increase in $CH(G)$ should imply an increase in error probability of the overall model.
Related Work	CH has not yet been formalized in literature.

The Control Flow Complexity metric was introduced in [Car05d] for measuring how difficult it is to consider all potential states after a split depending on its type.

Symbol	CFC
Definition	CFC is the sum over all connectors weighted by their potential combinations of states after the split.
Metric	$CFC(G) = \sum_{c \in S_{and}} 1 + \sum_{c \in S_{xor}} \lvert c_{xor} \bullet \rvert + \sum_{c \in S_{or}} 2^{\lvert c_{or} \bullet \rvert} - 1$
Rationale	A process model with a high CFC should be more likely to contain errors according to the above argument.
Limitations	Models with the same structure but with different connector labels may have a huge difference in CFC while they are equally easy to understand.
Hypothesis	An increase in $CFC(G)$ should imply an increase in error probability of the overall model.
Related Work	CFC is inspired by [McC76] and introduced in [Car05d].

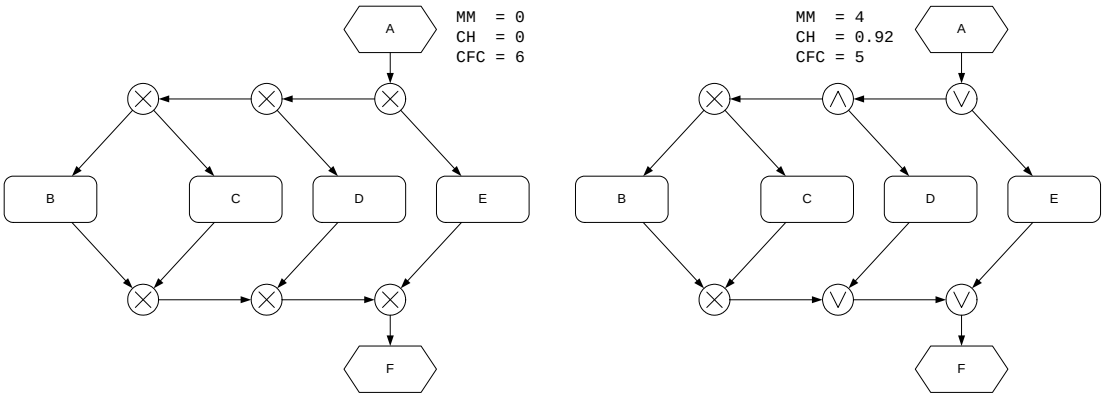


Figure 5.7: Two EPCs with different connector types

Figure 5.7 illustrates the three connector metrics by the help of two example EPCs. The EPC on the left-hand side has six XOR-connectors, each having a matching counterpart. Accordingly, the mismatch value is 0. Since there are only XOR-connectors, the heterogeneity value is also 0. Furthermore, the CFC value is 6, since each of the three split connectors represents a binary choice. The EPC on the right-hand side has the same structure, but partially different connector types. There is one AND-split and an OR-join that do not have a matching counterpart. This results in a mismatch value of 4. Moreover, three OR, two XOR, and one AND-connector yield a high heterogeneity value of 0.92. Finally, the CFC value is calculated by summing up 2 for the XOR-split and 3 for the OR-split which gives a result of 5.

5.5.5 Cyclicity

Cyclic parts of a model are presumably more difficult to understand than sequential parts. $|N_C|$ gives the number of nodes n_i for which a cycle exists such that $n_i \hookrightarrow n_i$, and cyclicity relates it to the total number of nodes.

Symbol	CYC
Definition	Cyclicity relates nodes on a cycle to the number of nodes.
Metric	$CYC_N = \|N_C\|/\|N\|$
Rationale	A process model with a high cyclicity should be more likely to contain errors. For a sequential model cyclicity is 0.

Limitations	There are models with the same cyclicity, but which differ in comprehensibility, e.g. if one is larger.
Hypothesis	An increase in $CYC(G)$ should imply an increase in error probability of the overall model.
Related Work	CYC has not yet been mentioned in literature. *Nissen* proposes to count the number of cycles instead [Nis98].

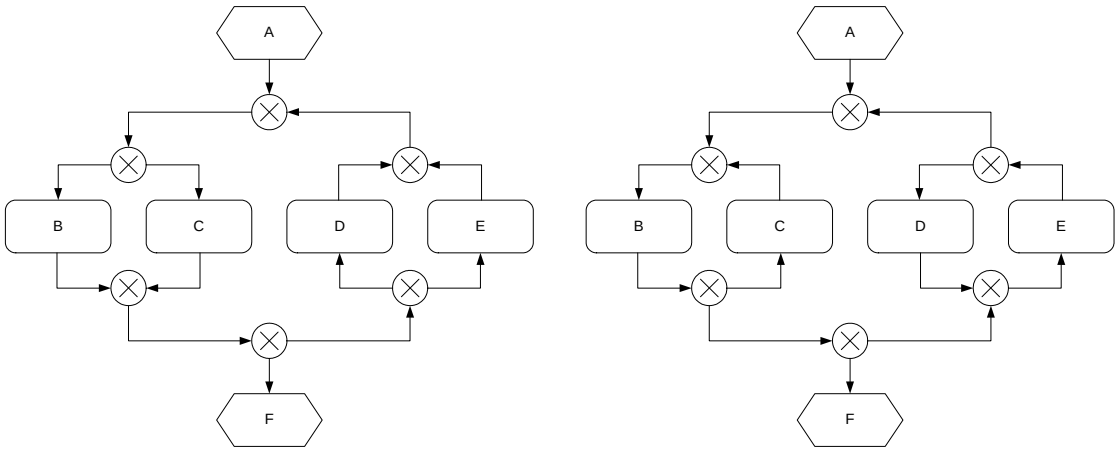


Figure 5.8: Two EPCs, one with nested cycles

Figure 5.8 depicts two similar EPCs. They have the same cyclicity since they have the same number of nodes that are on a loop. The difference is that the model on the right-hand side has two cycles that are nested in another cycle, while the EPC on the left side has two structured XOR-blocks on one loop. Since the cyclicity metric only captures how many nodes are on a cycle, it cannot distinguish between models with a different number of cycles.

5.5.6 Concurrency

Modelers have to keep track of concurrent paths that need to be synchronized. AND-splits and OR-splits introduce new threads of control, so that the number of control tokens potentially increases by the number of the output degree minus one. The Token Split

metric counts these newly introduced tokens. Concurrent tokens from the initial marking are not considered.

Symbol	TS
Definition	The token split sums up the output-degree of AND-joins and OR-joins minus one.
Metric	$TS(G) = \sum_{c \in C_{or} \cup C_{and}} d_{out}(n) - 1$
Rationale	A process model with a high token split value should be more likely to contain errors since it introduces a high degree of parallelism. A model with $TS = 0$ does not introduce new threads of execution after instantiation.[4]
Limitations	There are models with the same token split value, but which differ in comprehensibility, e.g. if one is structured.
Hypothesis	An increase in $TS(G)$ should imply an increase in error probability of the overall model.
Related Work	The maximum number of tokens was proposed by *Lee and Yoon* in [LY92]. While that approach is appealing, it is more computation intense that the token split metric. In *Nissen* [Nis98], the concept of parallelism is captured by an approximation as number of nodes divided by diameter assuming that all splits introduce parallelism. *Morasca* proposes concurrent contribution which is related to token splits [Mor99]. *Balasubramanian and Gupta* relate parallel nodes to nodes in total [BG05].

Figure 5.9 illustrates that the token split gives an upper bound for the number of tokens in a model assuming boundedness. The EPC on the left-hand side contains two blocks of concurrency is sequence. Therefore, its maximum number of tokens is lower than the token split value. In the model on the right-hand side, the two blocks are nested. In this case the maximum number of tokens matches the token split value.

The set of presented metrics reveals that there are several independent factors that

[4] Still there may be concurrency due to multiple start events.

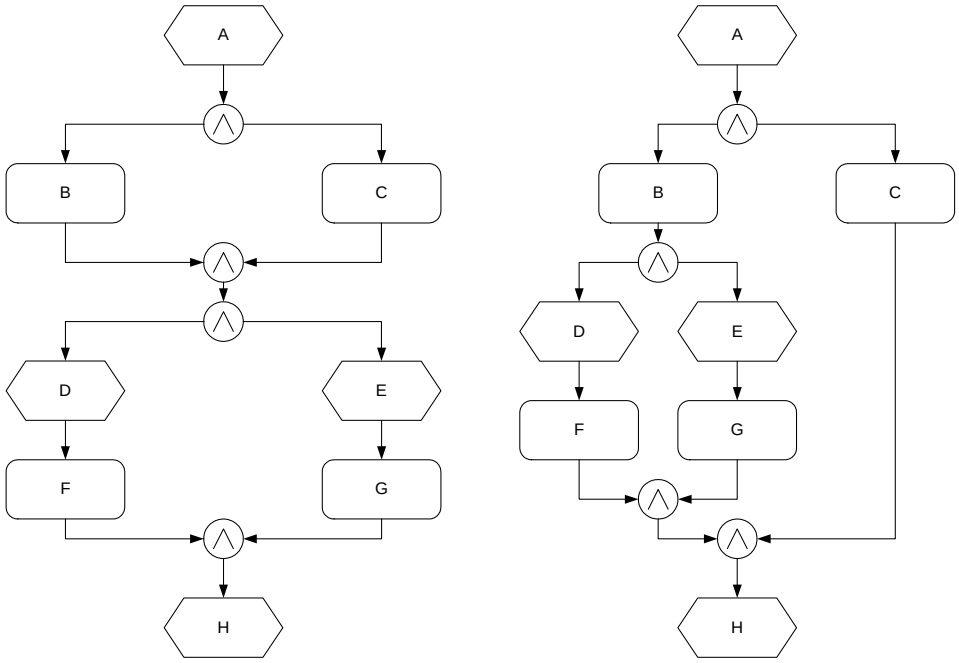


Figure 5.9: Two EPCs with the same token split values

presumably affect error probability. As a consequence, approaches trying to squeeze the complexity of a process model[5] into a single metric seem doomed to fail. A similar observation is made by *Zuse* [Zus91] who describes software complexity as a multi-dimensional phenomenon. In the subsequent section, we revisit the example EPC from Figure 3.1 to illustrate the metrics.

5.6 Calculating Error Metrics

In Section 3.1, we introduced the example of a loan request EPC taken from [NR02]. Table 5.1 summarizes the different metrics and Figure 5.10 illustrates which nodes and arcs contribute to the more elaborate metrics. Since the different count metrics, in particular for size, can be easily read from the model, we focus on those that need to be calculated

[5]The complexity of a process model can be defined as the degree to which a process model is difficult to analyze, understand, or explain (cf. [IEE90].

from the process graph, i.e. separability, sequentiality, structuredness, depth, cyclicity, and diameter.

Table 5.1: Metrics derived from the EPC example

Size S_N	27	density Δ	0.040
Size S_E	11	density CNC	1.037
Size S_{E_S}	1	av. connector degree $\overline{d_C}$	3
Size $S_{E_{Int}}$	8	max. connector degree $\widehat{d_C}$	3
Size S_{E_E}	2	separability Π	0.440
Size S_F	8	sequentiality Ξ	0.345
Size S_C	8	structuredness Φ	0.556
Size $S_{S_{XOR}}$	3	depth Λ	1
Size $S_{J_{XOR}}$	2	mismatch MM	8
Size $S_{S_{AND}}$	2	heterogeneity CH	0.710
Size $S_{J_{AND}}$	2	control flow complexity CFC	8
Size $S_{S_{OR}}$	0	cyclicity CYC	0.259
Size $S_{J_{OR}}$	1	token splits TS	2
Size S_A	29		
Size $diam$	14		

The *separability ratio* Π depends on the identification of cut vertices (i.e. articulation points), i.e. those nodes whose deletion breaks up the graph in two or more disconnected components. Figure 5.10 displays articulation points with a letter A written next to the top left-hand side of the node. For example, if the function "record loan request" is deleted, the start event is no longer connected with the rest of the process model. There are eleven articulation points in total yielding a separability ratio of $11/(27-2) = 0.440$. Note that start and end events do not belong to the set of articulation points, since their deletion does not increase the number of separate components.

The *sequentiality ratio* Ξ is calculated by relating the number of sequence arcs, i.e. arcs that connect functions and events, to the total number of arcs. Figure 5.10 highlights sequence arcs with an s label. There are ten sequence arcs and 29 arcs altogether which results in a sequentiality ratio of $10/29 = 0.345$. The degree of *structuredness* Φ relates the size of a reduced process model to the size of the original one. Figure 5.10 shows those elements with a cross on the left-hand side that are eliminated by reduction of trivial constructs. Other structured reduction rules are not applicable. Since 15 elements

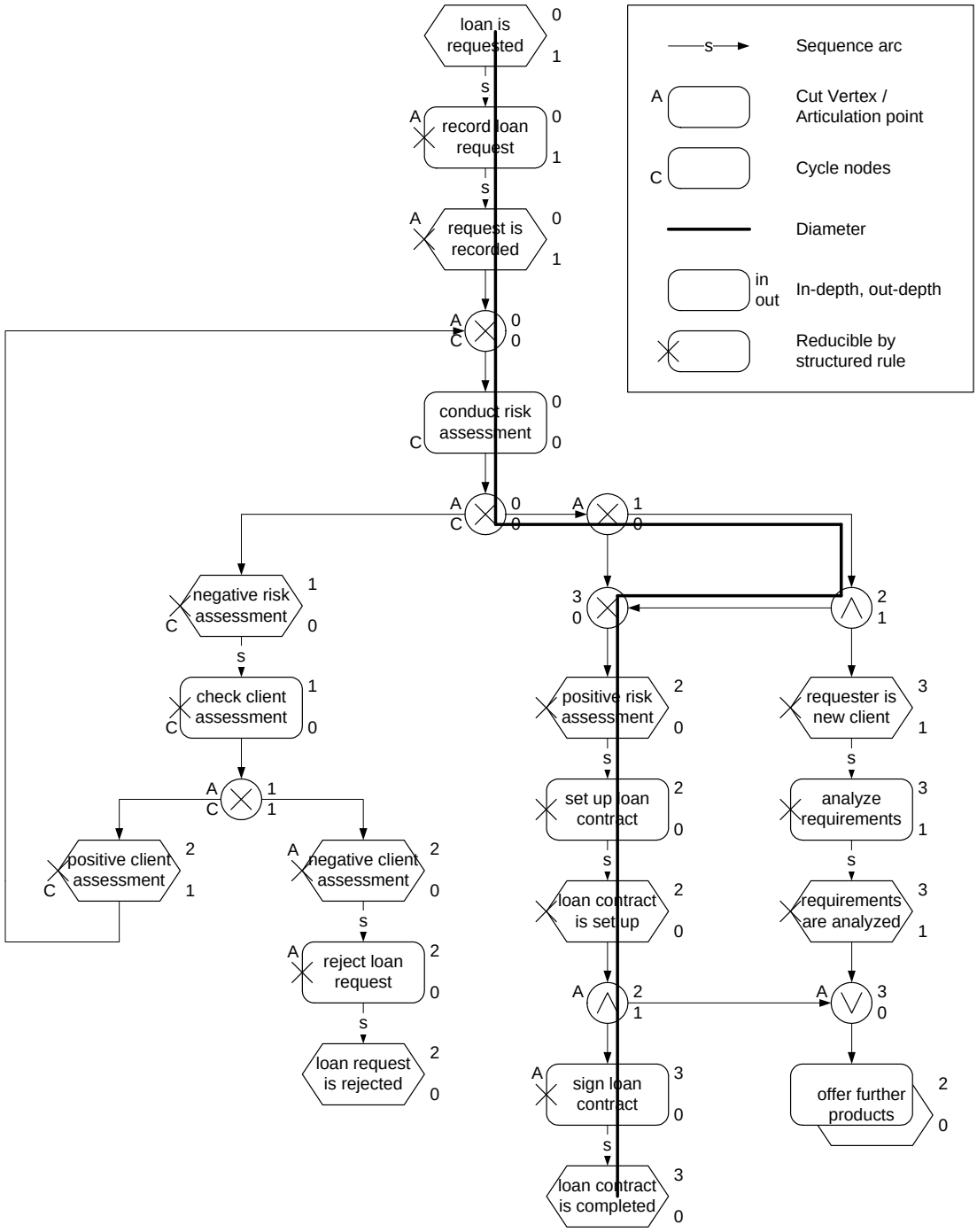


Figure 5.10: EPC example with sequence arcs, articulation points, cycle nodes, diameter, depth, and reducible nodes

are deleted by reduction, the structuredness ratio is $1 - 15/27 = 0.556$. The *in-depth and out-depth* is also indicated for each node in Figure 5.10. The depth of a node is then the minimum of in-depth and out-depth. Several nodes have a depth of 1, which is a maximum, and therefore also the depth of the overall process. The *cyclicity* is based on the relation between number of nodes on a cycle and nodes in total. Figure 5.10 shows nodes on a cycle with a letter C written to the left-hand side bottom. There are seven such nodes yielding a cyclicity ratio of $7/27 = 0.259$. Finally, Figure 5.10 connects those 14 nodes that are on the *diameter* with a bold line.

We implemented the calculation of the various metrics as an extension to *xoEPC* (see Section 4.3.2). For each EPC that is analyzed by the program, the whole set of metrics is calculated and written to the entry for the model in the errorreport.xml file. We will use this feature in the following chapter for the analysis of an extensive collection of EPC business process models from practice.

5.7 Summary

In this chapter, we discussed the suitability of error metrics from a theoretical point of view. Revisiting related research in the area of network analysis, software measurement, and metrics for business process models, we found that several aspects of process models were not yet combined in an overall measurement framework. Based on theoretical considerations, *we presented a set of 15 metrics related to size and 13 metrics that capture various aspects of the structure and the state space of the process model*. For each of the metrics, we discussed a plausible connection with error probability and formulated respective hypotheses. In the following chapter, we will test these hypotheses for a large set of EPC business process models from practice, using statistical methods.

Chapter 6

Validation of Error Metrics

In this chapter, we test the validity of metrics that were defined in the previous chapter for predicting errors in EPC business process models. In Section 6.1, we provide an overview of how the analysis data is generated. Subsequently, Section 6.2 describes the sample of EPCs from practice that we use for the analysis. In particular we discuss a disaggregation by the EPC model group and by error, as well as a correlation analysis between metrics and error. Based on this sample, we calculate a logistic regression model for predicting error probability with the metrics as input variables in Section 6.3. In Section 6.4, we then test the regression function for an independent sample of EPC models from textbooks as a cross-validation. Finally, Section 6.5 summarizes the findings.

6.1 Analysis Data Generation

Figure 6.1 gives an overview of the analysis data generation process. As input, we use four sets of EPC business process models that are available in the XML interchange

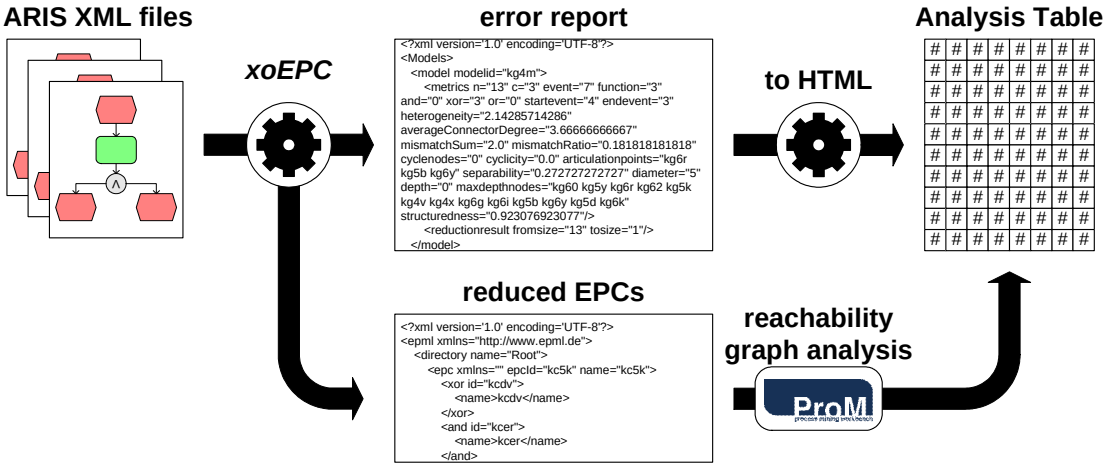


Figure 6.1: Overview of the analysis

format of ARIS Toolset of IDS Scheer AG. In Section 6.2 we describe these sets of EPC models in detail. As a first step, the set of ARIS XML files is read and processed by *xoEPC*, the batch program we introduced in Chapter 4. *xoEPC* applies the set of reduction rules that we described in Section 4.3.3, and generates an XML error report file that includes, among others, the following information for each EPC:

- processing time for the EPC,
- references to syntactical problems,
- references to errors,
- statistics about how often a certain reduction rule was applied
- information whether the model is reduced completely, and finally
- values for each of the metrics that we described in Section 5.5.

The error report XML file is then transformed to an HTML table by an XSLT program. Furthermore, each not completely reduced EPC is written to an EPML file. The reduced EPCs are then analyzed by the help of the reachability graph analysis plug-in for

ProM that we introduced in Section 4.2. The results of this analysis are added to the analysis table. Finally, the table is stored as an MS Excel file, since this format can be loaded by SPSS, the software package that we use for the statistical analysis. The complete list of variables of the analysis table is described in Appendix C.1.

6.2 The Sample of EPC Models

This section describes the sample of EPC models that we use for the validation of the set of metrics that we defined in the previous chapter. In particular, we present descriptive statistics disaggregated by group and error, as well as a correlation analysis between the variable *hasErrors* and each of the metrics.

The sample includes four collections of EPCs with a total of 2003 process models. All EPCs of the four groups were developed by practitioners.

1. *SAP Reference Model:* The first collection of EPCs is the SAP Reference Model. We already used this set of process models to illustrate the performance of the reduction rule verification approach, as implemented in *xoEPC* (see Chapter 4). The development of the SAP reference model started in 1992 and first models were presented at CEBIT'93 [KT98, p.VII]. Since then, it was developed further until version 4.6 of SAP R/3 which was released in 2000 [MADV06]. The SAP reference model includes 604 non-trivial EPCs.

2. *Service Model:* The second collection of EPCs stems from a German process reengineering project in the service sector. The project was conducted in the late 1990s. The modeling task was carried out by a project team with academic supervision. As an organization principle, the business processes were modelled in two separate groups, depending on whether they were supported by the ERP-system or not. The models that were defined in this project include 381 non-trivial EPCs. Furthermore, there are models describing the organization, the data, and information systems.

3. *Finance Model:* The third model collection contains the EPCs of a process documentation project in the Austrian financial industry. The project not only recorded the various business processes of the different functional units of the company, but also information systems, organization, and business forms. Altogether, it includes 935 EPC process models.

4. *Consulting Model:* The fourth collection covers a total of 83 EPCs from three different consulting companies. These companies reside in three different countries. The models of this collections also include organizational and functional models. The models are mainly used as reference models to support and streamline consulting activities of the companies.

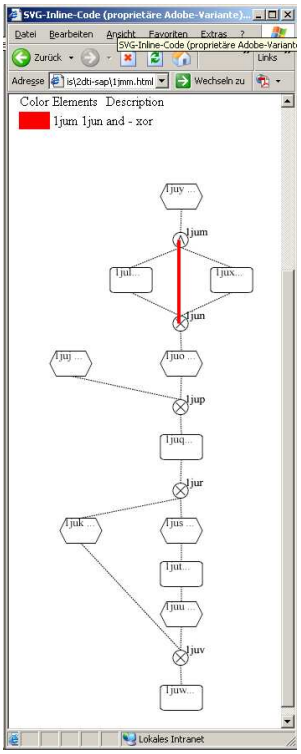


Figure 6.2: EPC with error from group 2

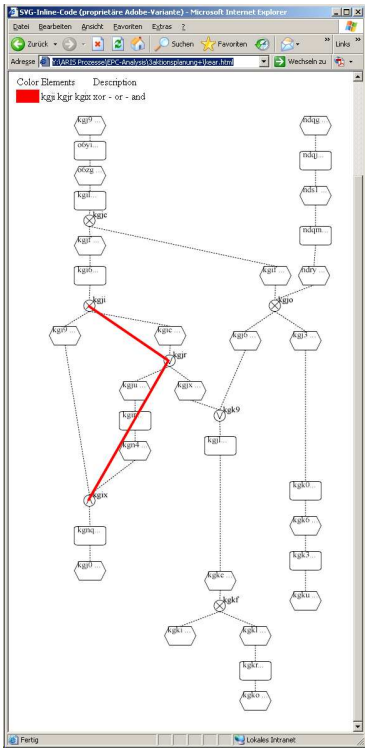


Figure 6.3: EPC with error from group 3

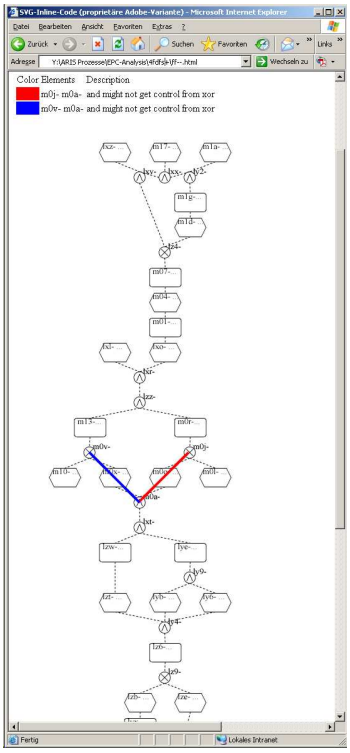


Figure 6.4: EPC with error from group 4

We will use the terms *group 1* synonymously for the SAP Reference Model, *group 2* for the Service Model, *group 3* for the Finance Model, and *group 4* for the Consulting

Model. Figures 6.2 to 6.4 show example EPCs with errors from the service model, the finance model, and the consulting model.

6.2.1 How do the four Groups differ?

In this section, we use descriptive statistics to characterize the overall EPC sample and its four sub-groups. In particular, we give mean values μ and standard deviation σ for each metric including size S_N and its variants, diameter $diam$, density Δ, coefficient of connectivity CNC, average and maximum connector degree $\overline{d_C}$ and $\widehat{d_C}$, Separability Π, Sequentiality Ξ, Structuredness Φ, Depth Λ, connector mismatch MM and heterogeneity CH, control flow complexity CFC, cyclicity CYC, and token splits TS.

Table 6.1: Mean and Standard Deviation of model sets disaggregated by group

Parameter	Complete Sample		SAP Ref. Model		Services Model		Finance Model		Consulting Model	
	μ	σ	μ_1	σ_1	μ_2	σ_2	μ_3	σ_3	μ_4	σ_4
S_N	20.71	16.84	20.74	18.74	22.14	15.71	19.50	15.30	27.64	21.35
S_E	10.47	8.66	11.50	10.44	9.54	7.80	9.93	7.41	13.20	9.99
S_{E_S}	2.43	2.70	3.87	3.84	1.99	2.02	1.59	1.12	3.43	3.38
S_{E_E}	2.77	3.20	4.49	4.78	2.35	2.08	1.77	1.24	3.39	3.12
S_F	5.98	4.94	4.03	3.81	8.23	5.45	6.22	4.79	7.22	5.95
S_C	4.27	5.01	5.21	6.22	4.37	4.20	3.35	3.90	7.22	6.96
$S_{C_{AND}}$	2.25	3.00	2.18	2.75	1.26	1.88	0.48	1.27	3.37	3.71
$S_{C_{XOR}}$	1.26	2.24	1.95	2.78	2.34	3.01	2.29	2.99	3.49	4.05
$S_{C_{OR}}$	0.76	1.54	1.08	1.81	0.77	1.27	0.57	1.45	0.35	0.89
$S_{J_{AND}}$	0.63	1.23	1.09	1.60	0.56	0.93	0.27	0.71	1.64	2.01
$S_{J_{XOR}}$	1.01	1.46	1.02	1.51	0.94	1.38	1.00	1.41	1.49	1.93
$S_{J_{OR}}$	0.37	0.82	0.46	1.03	0.37	0.69	0.32	0.73	0.22	0.54
$S_{S_{AND}}$	0.62	1.17	1.08	1.48	0.68	1.06	0.22	0.64	1.51	1.79
$S_{S_{XOR}}$	1.24	1.75	0.93	1.50	1.27	1.76	1.36	1.81	1.88	2.19
$S_{S_{OR}}$	0.37	0.86	0.62	1.14	0.33	0.57	0.25	0.74	0.13	0.46
S_A	21.11	18.87	20.80	20.84	22.50	17.43	20.12	17.54	28.14	22.80
$diam$	11.45	8.21	9.20	6.46	12.25	7.90	12.27	8.78	14.83	10.50
Δ	0.09	0.07	0.09	0.08	0.07	0.05	0.09	0.07	0.06	0.05
CNC	0.96	0.13	0.94	0.13	0.97	0.13	0.96	0.13	1.00	0.12
$\overline{d_C}$	3.56	2.40	3.30	1.46	3.12	1.66	2.50	1.62	3.08	0.81
$\widehat{d_C}$	2.88	1.60	4.36	2.72	3.85	2.30	2.91	2.11	3.81	1.22
Π	0.56	0.27	0.52	0.22	0.56	0.25	0.60	0.30	0.50	0.21
Ξ	0.46	0.31	0.29	0.28	0.45	0.26	0.59	0.29	0.32	0.22
Φ	0.88	0.11	0.83	0.14	0.90	0.09	0.90	0.08	0.81	0.17
Λ	0.70	0.74	0.55	0.71	0.79	0.66	0.72	0.76	1.00	0.86
MM	3.31	4.55	6.02	6.19	2.84	3.42	1.68	2.49	3.95	3.47
CH	0.28	0.35	0.43	0.38	0.31	0.34	0.15	0.28	0.38	0.34
CFC	382.62	8849.48	1187.98	16040.30	101.80	1678.82	10.17	52.79	6.71	7.35
CYC	0.01	0.08	0.02	0.09	0.04	0.13	0.00	0.01	0.02	0.09
TS	1.82	3.53	3.16	4.89	1.84	2.66	0.91	2.39	2.13	2.63

Table 6.1 gives an overview of the mean μ and the standard deviation σ for all met-

rics disaggregated by the four model groups. Several of the disaggregated mean values are quite close to each other, but in particular the Finance Model shows some striking differences. Firstly, it uses start and end events very scarcely (1.59 and 1.77 compared to 3.87 and 4.49 in average in the SAP Reference Model). Secondly, it has the highest mean in structuredness Φ and sequentiality Ξ. Figures 6.5 and 6.6 illustrate the distribution of both the latter metrics as box plots[1] disaggregated by group. In this type of diagram, the median is depicted as a horizontal line in a box that represents the interval between lower and upper quartile, i.e. the box refers to middle range EPCs ranked by the metric from 25% to 75%. Please note that the table indicates the *mean* while the box plot shows the *median*. The upper and lower lines outside the box define a one and a half multiple of the respective 25%–50% and 50%–75% quartiles. Values outside these two intervals are drawn as individual points and are considered to be outliers. From the two observations on start and end events, as well as on structuredness Φ and sequentiality Ξ, we might be tempted to conclude that the Finance Model contains the more structured EPCs, and thus might have less error models.

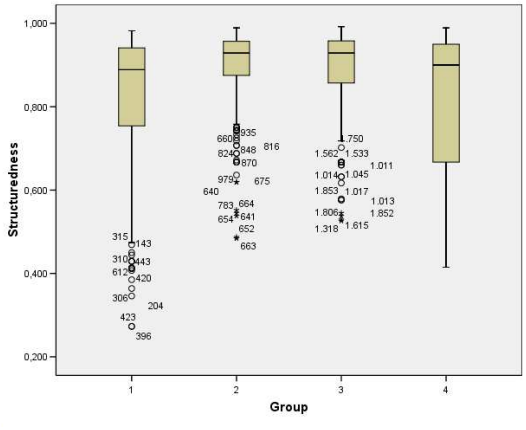


Figure 6.5: Box plot for structuredness Φ disaggregated by group

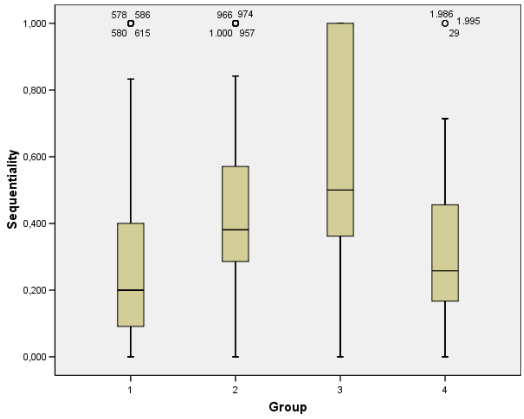


Figure 6.6: Box plot for sequentiality Ξ disaggregated by group

There is some evidence for such a hypothesis when we look at the number of errors in each of the four groups. Table 6.2 gives a respective overview. It can be seen that

[1]The box plot is particularly useful for exploratory data analysis. It was invented by *Tukey* in 1977 (see [Tuk77]). Box plots of all variables disaggregated by group are included in Appendix C.2.

there are 2003 EPCs in the overall sample with 215 of them having at least one error. Accordingly, there is an overall error ratio of 10.7%. 154 of the 215 errors were found by *xoEPC*. 156 EPCs could not be completely reduced and were analyzed with ProM. This analysis revealed that 115 of the unreduced EPCs still had errors. Please note that there are EPCs for which both *xoEPC* and ProM found errors. Therefore, the number of EPCs with errors is less than the sum of EPCs with *xoEPC* and ProM errors. The comparison of the groups shows that the error ratio is quite different. In the previous paragraph, we hypothesized that the finance model group might have less errors since it seems to follow certain guidelines that lead to more structured models. This might be an explanation for the low error ratio of only 3.3%. We might find some further evidence regarding the connection between metrics and errors in the subsequent section with a disaggregation by the boolean variable *hasErrors*.

Table 6.2: Errors in the sample models

Parameter Parameter	Complete Sample	SAP Ref. Model	Services Model	Finance Model	Consulting Models
xoEPC errors	154	90	28	26	10
Unreduced EPCs	156	103	18	17	18
ProM error EPCs	115	75	16	7	17
EPCs with errors	215	126	37	31	21
EPCs in total	2003	604	381	935	83
Error ratio	10.7%	20.9%	9.7%	3.3%	25.3%

6.2.2 How do correct and incorrect Models differ?

In this section, we discuss the distribution of the different metrics disaggregated by the variable *hasErrors*. Table 6.3 shows that there are quite large differences in the mean values of the EPC sub-sample with and without errors. It is interesting to note that the error mean μ_e is higher than the non-error mean μ_n for most metrics where we assumed a positive connection with error probability in Section 5.5, and smaller for those metrics with a presumably negative connection. The only case where this does not apply is the density metric. We discussed potential problems of this metric earlier in Chapter 5, and it seems that it works more accurately as a counter-indicator for size than as an indicator for

Table 6.3: Mean and Standard Deviation of the sample models disaggregated by error

Parameter	Complete Sample		Non-Error EPCs		Error EPCs		2 σ dev. up		2 σ dev. down	
	μ	σ	μ_n	σ_n	μ_e	σ_e	$\mu_n + 2\sigma_n$		$\mu_n - 2\sigma_n$	
S_N	20.71	16.84	18.04	13.48	42.97	24.08	44.99	$\approx \mu_e$		
S_E	10.47	8.66	9.06	6.69	22.17	13.19	22.45	$\approx \mu_e$		
S_{E_S}	2.43	2.70	2.04	2.04	5.69	4.65	6.12	$\approx \mu_e$		
S_{E_E}	2.77	3.20	2.25	2.11	7.02	6.19	6.47	$< \mu_e$		
S_F	5.98	4.94	5.67	4.65	8.53	6.33	14.97			
S_C	4.27	5.01	3.30	3.47	12.26	7.89	10.24	$< \mu_e$		
$S_{C_{AND}}$	2.25	3.00	0.85	1.47	4.74	3.89	3.78	$< \mu_e$		
$S_{C_{XOR}}$	1.26	2.24	1.85	2.60	5.50	3.97	7.05			
$S_{C_{OR}}$	0.76	1.54	0.60	1.33	2.02	2.35	3.27			
$S_{J_{AND}}$	0.63	1.23	0.40	0.81	2.54	2.12	2.02	$< \mu_e$		
$S_{J_{XOR}}$	1.01	1.46	0.82	1.24	2.63	2.06	3.29			
$S_{J_{OR}}$	0.37	0.82	0.32	0.74	0.79	1.26	1.79			
$S_{S_{AND}}$	0.62	1.17	0.44	0.84	2.13	2.09	2.12	$\approx \mu_e$		
$S_{S_{XOR}}$	1.24	1.75	1.04	1.56	2.86	2.31	4.16			
$S_{S_{OR}}$	0.37	0.86	0.27	0.68	1.22	1.51	1.63			
S_A	21.11	18.87	18.14	15.20	45.79	26.78	48.54	$\approx \mu_e$		
$diam$	11.45	8.21	10.63	7.71	18.25	9.01	26.06			
Δ	0.09	0.07	0.09	0.07	0.03	0.02	0.23			
CNC	0.96	0.13	0.95	0.13	1.05	0.08	1.21			
$\overline{d_C}$	3.56	2.40	2.80	1.66	3.57	0.68	6.11			
$\widehat{d_C}$	2.88	1.60	3.31	2.28	5.64	2.41	7.87			
Π	0.56	0.27	0.59	0.27	0.35	0.13			0.06	
Ξ	0.46	0.31	0.49	0.30	0.18	0.14			-0.12	
Φ	0.88	0.11	0.90	0.09	0.70	0.16			0.72	$> \mu_e$
Λ	0.70	0.74	0.61	0.69	1.45	0.73	1.98			
MM	3.31	4.55	2.54	3.45	9.71	6.92	9.44	$< \mu_e$		
CH	0.28	0.35	0.22	0.32	0.75	0.19	0.85			
CFC	382.62	8849.48	202.19	6306.23	1883.17	19950.26	12814.64			
CYC	0.01	0.08	0.01	0.06	0.07	0.17	0.12			
TS	1.82	3.53	1.28	2.46	6.26	6.62	6.20	$< \mu_e$		

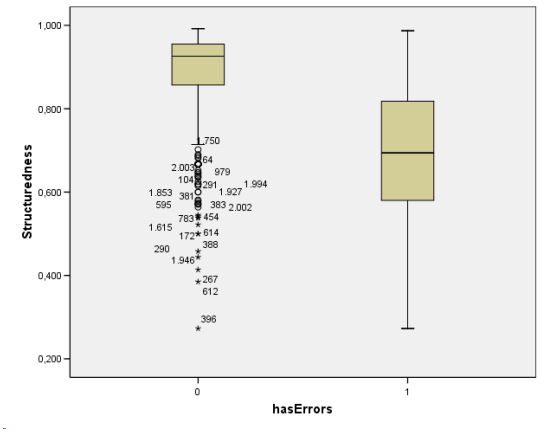


Figure 6.7: Box plot for structuredness Φ disaggregated by error

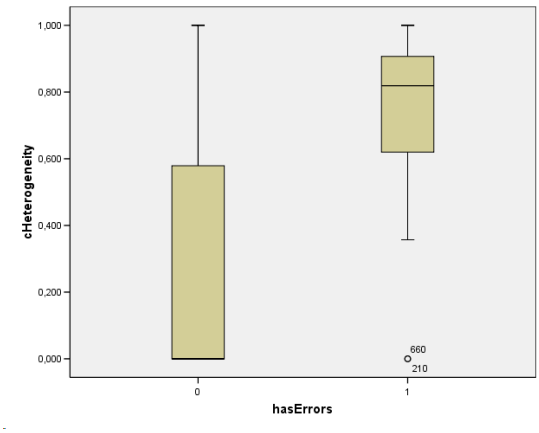


Figure 6.8: Box plot for connector heterogeneity CH disaggregated by error

the density of connections in the model. Indeed, there is a correlation of -0.659 between density and size. The two columns on the right hand side of Table 6.3 might provide the basis for proposing potential error thresholds. The first of these columns gives a double σ_n deviation upwards from the non-error mean μ_n. Assuming a normal distribution, only 2.5% of the population can be expected to have a metric value greater than this. The comparison of this value with the mean μ_e of the error EPCs gives an idea how well the two sub-samples can be separated by the metric. In several cases, the mean μ_e is outside the double σ_n interval around μ_n. The box plots in Figures 6.7 and 6.8 illustrate the different distributions. It can be seen that correct EPCs tend to have much higher structuredness values and lower connector heterogeneity values.[2] Box plots of all variables disaggregated by error can be found in the Appendix C.3. We verified the significance of the difference between the mean values by applying an analysis of variance (ANOVA). The result of the Kolmogorov-Smirnov test shows that the prerequisite of an approximative normal distribution is fulfilled by all variables (see Table C.3 on page 422). Furthermore, the F-statistic values of the analysis of variance indicates that the mean differences are significant with 99.9% confidence for all metrics (see Table C.4 on page 423). In the subsequent section, we gather further evidence regarding the direction of the connection between metrics and errors based on a correlation analysis.

6.2.3 Correlation Analysis

This section approaches the connection between error probability and metrics with a correlation analysis. In the Appendix C.5, we list the complete correlation table calculated according to *Pearson* for interval scale data and to *Spearman* for ordinal scale data. The tendency is the same for both methods. As a confirmation of the previous observation, all variables have the expected direction of influence, except for the density metric. Table 6.4 presents the Spearman correlation between *hasErrors* and the metrics ordered by strength of correlation. It can be seen that several correlations are quite considerable with absolute values between 0.30 and 0.50. The significance of all correlations is good with more than 99% confidence.

[2]The two outliners in Figure 6.8 are cyclic models with only OR-connectors in the first case and only AND-connectors in the second case.

Table 6.4: Spearman correlation between hasError and metrics ordered by absolute correlation

	hasError		hasError		hasError
$S_{J_{AND}}$	0,48	S_{E_E}	0,38	$S_{S_{OR}}$	0,31
CH	0,46	Δ	-0,37	$S_{S_{XOR}}$	0,31
$S_{C_{AND}}$	0,45	$S_{S_{AND}}$	0,37	CYC	0,30
S_C	0,43	Φ	-0,36	$S_{C_{OR}}$	0,30
MM	0,42	Ξ	-0,35	$diam$	0,30
CFC	0,39	$S_{C_{XOR}}$	0,35	Π	-0,29
S_A	0,38	S_{E_S}	0,35	CNC	0,28
TS	0,38	Λ	0,34	$\overline{d_C}$	0,23
S_N	0,38	$S_{J_{XOR}}$	0,33	S_F	0,19
S_E	0,38	$\widehat{d_C}$	0,33	$S_{J_{OR}}$	0,15

The ability of a metric to separate error from non-error models by ranking is illustrated in Figures 6.9 and 6.10. For Figure 6.9, all models are ranked according to their size. A point (x, y) in the graph relates a size x to the relative frequency of error models in a subset of models that have at least size x, i.e. $y = |\{\frac{errorEPC}{EPC} \mid S_N(EPC) > x\}|$. It can be seen that the relative frequency of error EPCs increases by increasing the minimum number of nodes. In particular, the relative frequency of error EPCs is higher than 50% for all EPCs of at least 48 nodes. In Figure 6.10 all models are ranked according to their structuredness, and (x, y) relates the structuredness x to the subset of models that have at most structuredness x. Here, the graph decreases and drops below 50% at a structuredness value of 0.80. Similar observations can be made for some of the other metrics, too. Altogether, the relative frequency of error models above 50% is reached if

number of nodes $S_N > 48$
number of connectors $S_C > 8$
number of events $S_E > 22$
number of end events $S_{E_e} > 7$
number of functions $S_F > 40$
number of arcs $S_A > 62$
token splits $TS > 7$
connector mismatch $MM > 9$
structuredness $\Phi < 0.8$

In this section, we have gathered some evidence that the hypothetical connections between metrics and error probability as postulated in Section 5.5 might actually hold. First, we have found considerable and statistically significant differences in the metrics'

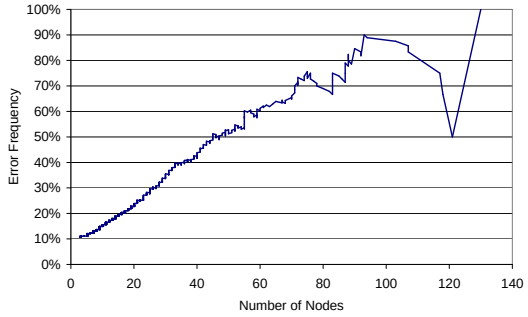


Figure 6.9: Error frequency to ordered number of nodes

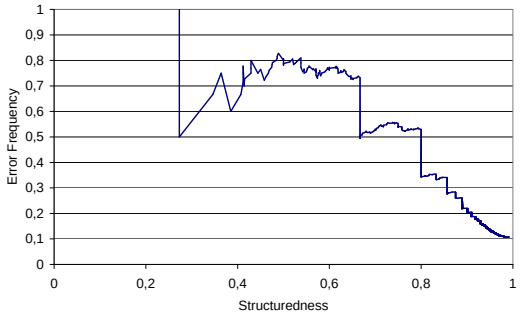


Figure 6.10: Error frequency to ordered structuredness

mean values for the sub-samples of EPCs with and without errors. The mean values of error EPCs, in particular, tend to be larger or smaller as expected by the hypotheses. Furthermore, correlation analysis confirmed the hypothetical direction of the connection between metrics and errors. The only exception is the density metric. It seems that this metric might be more suitable as a counter-indicator for size, than as an indicator of the relative number of arcs in the EPCs. Still, it must be kept in mind that correlation alone does not provide a means to predict error probability. In contrast to that, logistic regression allows a precise prediction by estimating the parameters of the logistic function. Therefore, we will investigate logistic regression in the following section.

6.3 Logistic Regression

This section provides an introduction to logistic regression analysis, and presents the result of its application for estimating the prediction model for error probability based on metrics.

6.3.1 Introduction to Logistic Regression

Logistic regression is a statistical model designed to estimate binary choices. It is perfectly suited to deal with dependent variables such as *hasErrors* with its range *error* and

no error. The idea of binary choice models is to describe the probability of a binary event by its odds, i.e., the ratio of event probability divided by non-event probability. In the *logistic regression* (or *logit*) model, the odds are defined as $logit(p_i) = ln(\frac{p_i}{1-p_i}) = \beta_0 + \beta_1 x_{1,i} + \cdots + \beta_k x_{k,i}$ for k input variables and i observations, i.e. EPC i in our context. From this follows that

$$p_i = \frac{e^{\beta_0 + \beta_1 x_{1,i} + \cdots + \beta_k x_{k,i}}}{1 + e^{\beta_0 + \beta_1 x_{1,i} + \cdots + \beta_k x_{k,i}}}$$

The relationship between input and dependent variables is represented by an S-shaped curve of the logistic function that converges to 0 for $-\infty$ and to 1 for ∞ (see Figure 6.11). The cut value of 0.5 defines whether event or non-event is predicted. $Exp(\beta_k)$ gives the multiplicative change of the odds if the input variable β_k is increased by one unit, i.e. $Exp(\beta_k) > 1$ increases and $Exp(\beta_k) < 1$ decreases error probability. The actual value $Exp(\beta_k)$ cannot be interpreted in isolation since its impact depends upon the position on the non-linear curve [JHG+88, p.791].

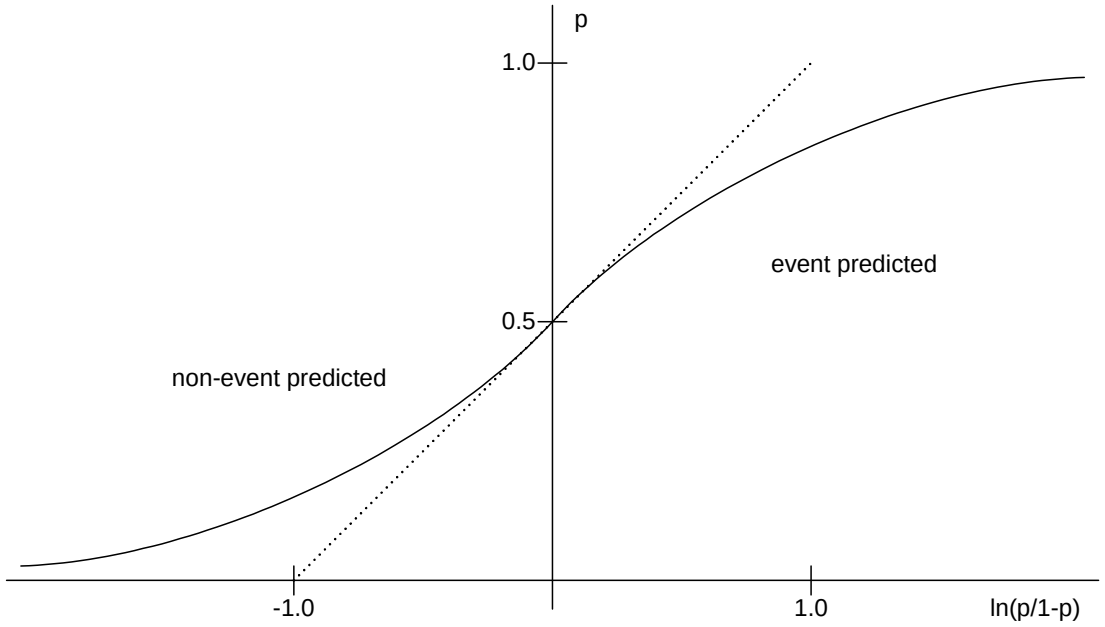


Figure 6.11: S-shaped curve of the logistic regression model

The significance of the overall model is assessed by the help of two statistics. Firstly, the *Hosmer & Lemeshow* Test should be greater than 5% to indicate a good fit based

on the difference between observed and predicted frequencies (cf. [HL00]). Secondly, *Nagelkerke's* R^2 ranging from 0 to 1 serves as a coefficient of determination indicating which fraction of the variability is explained [Nag91]. Furthermore, each estimated coefficient of the logit model is tested using the *Wald* statistic, for being significantly different from zero. The significance should be less than 5%. We calculate the logistic regression model based on a stepwise introduction of those variables that provide the greatest increase in likelihood. For more details on logistic regression, see [HL00, HATB98, BEPW03].

6.3.2 Preparatory Analyses

Before calculating a multivariate logistic regression model for error probability, we carry out two preparatory analyses. First, we check collinearity, then we determine which variables are included in the regression model. Furthermore, we exclude 29 EPCs from the analysis that are not relaxed syntactically correct. While it is possible to find errors in these models (as we did in Section 4.3.4), it is not appropriate to use them in a regression analysis for predicting errors in EPCs that fulfill relaxed syntactical correctness.[3]

Collinearity describes the phenomenon that at least one of the independent variables can be represented as a linear combination of other variables. The absence of collinearity is not a hard criterion for the applicability of logistic regression, but it is desirable. In a variable set without collinearity, every variable has a so-called tolerance value[4] higher than 0.1, otherwise there is a collinearity problem. In the original variable set (Table D.1) there are several collinearity problems. We dropped all count metrics, apart from S_N, since they were highly correlated. This resulted in a reduced variable set with almost no collinearity problems (Table D.2). The S_N metric is close to the 0.1 threshold and therefore kept in the metrics set.

As a second step, we calculated univariate models with and without a constant in order to check whether all inputs, i.e. the constant and each metric, were significantly

[3] Still, the effect of this choice is minimal. Including the 29 EPCs with syntax problems yields a logistic regression model with the same metrics and similar coefficients, and the same *Nagelkerke* R^2.

[4] The tolerance value is calculated based on the variance inflation factor. For an overview of multicollinearity detection methods refer to [JHG+88, Ch.21].

different from zero (see Tables D.3 and D.4). As a conclusion from these models we drop the constant and the control flow complexity CFC for the multivariate analysis. Firstly, the constant is not significantly different from zero (Wald statistic of 0.872 and 0.117) in the separability and the sequentiality model which suggests that it is not necessary. Secondly, the CFC metric is not significantly different from zero (Wald statistic of 0.531 and 0.382) in both models with and without constant. All other metrics stay in the set of input variables from the multivariate logistic regression model.

6.3.3 Multivariate Logistic Regression Model

This section presents the results of the multivariate logistic regression model. We use a stepwise introduction of the variables to the logit model selected for its potential to improve the likelihood. Variables are only introduced if their *Wald* statistic is better than 0.05, and they are excluded if this statistic becomes worse than 0.1. Such a stepwise approach for determining the best set of variables is appropriate in particular for a setting where little is known about the relative importance of the variables (cf. [HL00, p.116]). The final model was calculated in nine steps and includes seven variables. Figure D.4 in the Appendix gives an overview of the variables that were introduced in each step. It is interesting to note that the hypothetical impact direction of the included metrics is reconfirmed. All variables have an excellent *Wald* statistic value (better than 0.001) indicating that they are significantly different from zero. Furthermore, the *Hosmer & Lemeshow* test is greater than 0.05, which is also a good value. Finally, the *Nagelkerke* R^2 has an excellent value of 0.901 indicating a high degree of explanation (cf. Figure D.2 in the Appendix).

Based on the regression results, we can derive a classification function $p(EPC)$ for EPCs. It predicts that the EPC has errors if the result is greater than 0.5. Otherwise it predicts that there are no errors in the EPC. It is calculated by the help of the metrics coefficient of connectivity CNC, connector mismatch MM, cyclicity CYC, separability Π, structuredness Φ, connector heterogeneity CH, and the diameter $diam$.

$$p(EPC) = \frac{e^{logit(EPC)}}{1 + e^{logit(EPC)}}$$

with

$$\begin{aligned} logit(EPC) = \quad & +4.008\, CNC \\ & +0.094\, MM \\ & +3.409\, CYC \\ & -2.338\, \Pi \\ & -9.957\, \Phi \\ & +3.003\, CH \\ & +0.064\, diam \end{aligned}$$

It is easy to calculate an error prediction for an EPC based on this function. Figure D.3 in the Appendix shows the classification tables of all nine steps. It can be seen that

- 1724 EPCs are correctly predicted to have no errors,
- 155 EPCs are correctly predicted to have errors,
- 58 EPCs are predicted to have no errors, but actually have errors, and
- 37 EPCs are predicted to have errors, but actually have none.

Altogether 1879 EPCs have the correct prediction. The overall percentage is 95.2%, that is 6% better than the naive model that always predicts no error (89.2%). Furthermore, there are 213 EPCs with errors in the reduced sample. 155 of them are correctly predicted, i.e. 72.7%. Finally, the prediction function gives a clue about the relative importance of the different metrics. Structuredness Φ appears to be the most important parameter since its absolute value is three times as high as the second. Likewise, the coefficient of connectivity CNC, cyclicity CYC, separability Π, and connector heterogeneity CH seem to be of comparable importance. Finally, connector mismatch MM and the diameter $diam$ might be of minor importance.

After this, we excluded the metrics of the regression model and calculated a second best model without the coefficient of network connectivity CNC, connector mismatch MM, cyclicity CYC, separability Π, structuredness Φ, connector heterogeneity CH, and without the diameter $diam$. The idea is to gain insight into the direction of the

influence of further metrics on error probability. This second best model includes sequentiality Ξ, density Δ, and size S_N. The *Hosmer & Lemeshow* Test fails to indicate a good fit since the value is less than 5% after the second step. The value of *Nagelkerke's* R^2 still indicates a high fraction of explanation of the variability with a value of 0.824 and 91.4%, if all cases are classified correctly (cf. Appendix D.4). These figures indicate that the second best regression model is less powerful than the first model. The estimated equation is

$$logit_2(EPC) = -6.540\ \Xi - 23.873\ \Delta + 0.034\ S_N$$

After that, we also excluded the metrics of the second best regression model, i.e. only token split TS, average and maximum connector degree $\overline{d_C}$ and $\widehat{d_C}$, and Depth Λ were considered. Again, the *Hosmer & Lemeshow* Test fails to indicate a good fit since the value is less than 5% and the *Nagelkerke's* R^2 reaches "only" (compared to the previous models) a value of 0.627. Furthermore, 72.9% of all cases are classified correctly indicating a weaker capability to predict errors correctly, compared to the other models (cf. Appendix D.5). The estimated equation is

$$logit_3(EPC) = 0.194\ TS - 1.371\ \overline{d_C} + 0.405\ \widehat{d_C} + 0.440\ \Lambda$$

It is interesting to note that most coefficients of the different regression models confirm the expected direction of influence on error probability. Beyond that, two variables have an impact opposite to the expectation: the density Δ and the average connector degree $\overline{d_C}$. We already identified potential problems with density in Sections 5.2 and 5.5.2. It appears that this metric is stronger negatively connected with size than with the degree of connections in the process model. In contrast to that, the unexpected sign of the coefficient for average connector degree $\overline{d_C}$ seems to be due to a positive correlation with structuredness of 0.251 which is significant at a 99% confidence level. Since structuredness is not included in the variable set of the third best regression model, the average connector degree $\overline{d_C}$ apparently captures some of its negative impact on error probability.

In the following section, we analyze how well the different regression function is able to forecast errors in a sample of EPCs that was not included in the estimation.

6.4 External Validation

In this section, we utilize the estimated function to predict errors in EPCs from a *hold-out sample*. This external validation step is of paramount importance for establishing the criterion validity of the measurements, i.e. their pragmatic value (cf. Section 5.1) to demonstrate that the model is not overfitting the data, and thus, can also be used to predict errors in other model samples (cf. [HL00, pp.186]). Basically, a holdout sample is only one option for external validation. There are several techniques for *cross-validation* in which the original sample is partitioned into a training set and a test set. In k-fold cross-validation, the data set is split into k mutually exclusive subsets that each serve as a test set for an estimation that is calculated using the respective rest set. In the leave-one-out case (also called jackknife), only one case is left out and it is used to validate the estimation done with the rest. For a sample size N, this procedure is repeated N/k times, i.e. N times for the jackknife method. Another technique is *bootstrapping*. In the validation phase, the bootstrap sample is built by sampling n instances from the data with replacement. Several papers compare the three validation methods theoretically and by running simulations (cf. e.g. [Sto74, Gon86, Koh95]). Cross-validation and bootstrapping are important, in particular, when the sample size is small relative to the number of parameters, e.g. for 19 independent variables with 155 observations, as discussed in [Gon86]. Since our sample size is more than 100 times as large as the number of input variables (15 metrics without collinearity to 2003 EPC models), we deem it justified to consider an independent holdout sample and disregard cross-validation and bootstrapping.

For testing the performance of the prediction function, we gathered a holdout sample from popular German EPC business process modeling textbooks. The sample includes 113 models from the following books in alphabetical order:

- Becker and Schütte: *Handelsinformationssysteme*, 2004 [BS04]. This book discusses information systems in the retail sector with a special focus on conceptual modeling. In particular, it covers 65 EPC models that we include in the holdout sample.
- Scheer: *Wirtschaftsinformatik: Referenzmodelle für industrielle Geschäfts-*

prozesse, 1998 [Sch98b]. This textbook is an introduction to the ARIS framework and uses reference models for production companies to illustrate it. We include 27 EPC reference models in the holdout sample from this book.

- Seidlmeier: *Prozessmodellierung mit ARIS*, 2002 [Sei02]. This book is another introduction to the ARIS framework. It features 10 EPCs that we include in the holdout sample.

- Staud: *Geschäftsprozessanalyse: Ereignisgesteuerte Prozessketten und Objektorientierte Geschäftsprozessmodellierung für Betriebswirtschaftliche Standardsoftware*, 2006 [Sta06]. This book focuses on business process modeling and EPCs in particular. We include 13 EPCs from this book in the holdout sample.[5]

Classification Table

		Predicted		
		hasErrors		Percentage
Observed		0	1	Correct
hasErrors	0	86	2	97,73%
	1	9	16	64,00%
Overall Percentage				90,27%

The cut value is ,500
113 cases included

Figure 6.12: Classification table for EPCs from the holdout sample

All EPCs in the holdout sample were checked for errors, first with *xoEPC* and afterwards with the ProM plug-in if the rest size was greater than two. Altogether there are 25 of the 113 models that have errors, i.e. 21.43%. Based on the metrics generated by *xoEPC*, we can easily apply the prediction function. The result of this calculation is summarized in the classification table in Figure 6.12. It can be seen that 102 of the 113 EPCs are classified correctly, i.e. 86 models without errors are predicted to have none and 16 with errors are predicted to have at least one. Altogether 90.27% of the 113 EPCs were predicted correctly (81.25% with the second best model and 78.57% with the third best model). Please note that there is a difference between the interpretation of this classification result and the one in Section 6.3.3. During the estimation of the logistic regression

[5]The largest EPC of this book has 288 nodes and required 113 minutes processing time.

the sample is known. Therefore, the lowest possible classification result is defined by predicting no error for every EPC which would yield a correct prediction 89.2% of the cases for the sample in Section 6.3.3. Accordingly, the classification result of applying the estimated function on the estimation sample must be compared to this trivial classification. In Section 6.3.3 the regression function improves the result from 89.2% to 95.2%. Here, we used a given function to classify an independent sample. The lowest possible classification result in this setting is 0%, while 50% might be expected for a random function. Using the regression function for the independent sample increases the classification result from 50% to 90.27%.

Based on the De Moivre-Laplace theorem, we are also able to calculate a confidence interval for the accuracy of the prediction function. Using Equation 3 of [Koh95] with a confidence value of 95% yields an accuracy interval from 81.15% to 96.77%, i.e. the prediction can be expected to be correct in at least 81.15% of the cases with a 95% confidence. Therefore, this result strongly supports the validity of the regression function (and also the second and the third best model) for predicting error probability.

6.5 Summary

In this section, we conducted several statistical analyses related to the hypotheses on a connection between metrics and error probability. The results strongly support the hypotheses since the mean difference between error and non-error models, the correlation coefficients, and the regression coefficients confirm the hypothetical impact direction of all metrics except the density metric Δ (see Table 6.5) and partially the average connector degree $\overline{d_C}$. The density metric appears to be more closely related to the inverse of size than the relative number of arcs of an EPC. The wrong sign of the average connector degree in the regression model seems to be caused by a positive correlation with structuredness Φ, which tends to reduce error probability.

Table 6.5: Hypothetical and empirical connection between metrics and errors

	Hypothetical connection	$\mu_e - \mu_n$	Correlation	Regression coefficient	Direction
S_N	+	24.93	0.38	0.034[b)]	confirmed
S_E	+	13.11	0.38		confirmed
S_{E_S}	+	3.65	0.35		confirmed
S_{E_E}	+	4.76	0.38		confirmed
S_F	+	2.86	0.19		confirmed
S_C	+	8.96	0.43		confirmed
$S_{C_{AND}}$	+	3.89	0.45		confirmed
$S_{C_{XOR}}$	+	3.65	0.35		confirmed
$S_{C_{OR}}$	+	1.41	0.30		confirmed
$S_{J_{AND}}$	+	2.14	0.48		confirmed
$S_{J_{XOR}}$	+	1.81	0.33		confirmed
$S_{J_{OR}}$	+	0.47	0.15		confirmed
$S_{S_{AND}}$	+	1.70	0.37		confirmed
$S_{S_{XOR}}$	+	1.82	0.31		confirmed
$S_{S_{OR}}$	+	0.95	0.31		confirmed
S_A	+	27.64	0.38		confirmed
$diam$	+	7.62	0.30	0.064[a)]	confirmed
Δ	+	-0.06	-0.37	-23.873[b)]	not confirmed
CNC	+	0.11	0.28	4.008[a)]	confirmed
$\overline{d_C}$	+	0.76	0.23	-1.371[c)]	partially confirmed
$\widehat{d_C}$	+	2.33	0.33	0.405[c)]	confirmed
Π	-	-0.24	-0.29	-2.338[a)]	confirmed
Ξ	-	-0.31	-0.35	-6.540[b)]	confirmed
Φ	-	-0.20	-0.36	-9.957[a)]	confirmed
Λ	+	0.85	0.34	0.440[c)]	confirmed
MM	+	7.18	0.42	0.094[a)]	confirmed
CH	+	0.54	0.46	3.003[a)]	confirmed
CFC	+	1680.99	0.39		confirmed
CYC	+	0.06	0.30	3.409[a)]	confirmed
TS	+	4.97	0.38	0.194[c)]	confirmed

[a)] first regression model, [b)] second best, [c)] third best

Chapter 7

Conclusions

This chapter discusses the results of this thesis. In Section 7.1 we summarize the main results, and in Section 7.2 we discuss the implications of these results. Finally, Section 7.3 identifies some open questions for future research.

7.1 Summary of the Results

This doctoral thesis presented a novel holistic approach for the verification and prediction of errors in EPC business process models. Against the state of the art, the technical results of this thesis can be summarized as follows.

Formalization of the OR-join: We show that existing formalizations of the OR-join suffer either from a restriction of the EPC syntax or from non-intuitive behavior. In Chapter 3, we present a novel formalization of EPC semantics including OR-joins that is applicable for any EPC that is relaxed syntactically correct and provides

intuitive semantics for blocks of matching OR-splits and OR-joins. The calculation of the reachability graph is implemented as a plug-in for ProM as a proof of concept.

Verification of process models with OR-joins and multiple start and end events: Verification techniques for EPC process models with OR-joins and multiple start and end events suffer from the problem that they either use an approximation of the actual behavior, that they build on non-intuitive semantics, or that they are not tailored to cope with multiple start and end events. In Chapter 4 of this thesis, we specify a dedicated soundness criterion for EPC business process models with OR-joins and multiple start and end events. Furthermore, we define two verification approaches for EPC soundness, one as an explicit analysis of the reachability graph, and a second based on reduction rules to provide a better verification performance. Both approaches are implemented as a proof of concept.

Metrics for business process models: Metrics play an important role in operationalizing various quality-related aspects of business process models. While the current state of the art in business process model measurement is mainly inspired by software metrics and hardly consolidated, Chapter 5 provides an extensive overview of existing work. Furthermore, we introduce new metrics that capture important process model concepts such as partionability, connector interplay, cyclicity, and concurrency, and discuss their theoretical connection with error probability.

Validation of metrics as error predictors: Up to now there is little empirical evidence for the validity of business process model metrics as predictors for error probability. In Chapter 6 of this thesis, we use statistical methods to confirm the hypothetical connection between metrics and errors. Furthermore, we use logistic regression to estimate a error prediction function. This function not only fits an extensive EPC sample nicely, but also shows a good performance in terms of external validity to predict errors in an independent EPC sample.

7.2 Discussion

The results of this thesis have some more general implications for business process modeling. On the following pages we discuss the implications for 1) the importance of verification in practice, 2) for improvements of the business process modeling process, 3) for future business process modeling tools, and 4) for teaching of business process modeling languages.

1. *Importance of Verification:* The amount of errors in the different EPC model collections from practice that we used in this thesis emphasizes the importance of verification. We showed that an error ratio of about 10% is the average over the samples, with 3.3% being the minimum. While verification has been discussed for some time, this thesis demonstrates that the different techniques have matured to handle large sets of several thousand business process models on a common desktop computer. This observation relates to the gap between business analysis and information systems engineering in business process modeling (see [STA05, p.141] or [HR07, pp.424]), i.e. the refusal of engineers to reuse process documentations for systems implementation. While this gap is frequently accepted as a natural breach, this thesis tells a different story: the considerable amount of formal errors in documentation models makes it hardly possible to directly reuse them in the implementation. Therefore, the utilization of verification techniques in practice might be the key to eventually close this gap in the future.

2. *Business Process Modeling Process:* In this thesis, we gathered substantial theoretical and statistical evidence that formal errors are connected with several characteristics of the process model. This finding provides the opportunity to use process model metrics for the management of the design process and of process model quality. This is in particular the case if different design options have to be evaluated, and one of multiple models might be considered to be superior regarding error-probability based on some metric. Furthermore, the strong connection between size and errors offers objective input for decisions regarding the question when a model should be split up or when model parts should be put in a subprocess. There is clearly a need for further research in this area. Nevertheless, our

findings represent a major step towards establishing business process modeling as an engineering discipline beyond the intuition of the modeler.

3. *Business Process Modeling Tools:* Both items 1) and 2) call for respective tool support. While the verification techniques are apparently mature enough to deal with large models from practice, there seems to be too little attention paid to this issue by tool vendors. Indeed, tool vendors should have an interest in these topics since the lack of respective features has a negative impact on the productivity of the business process modeling exercise: models cannot be reused for system development, business users cannot interpret the models properly, and conclusions can hardly be drawn from the models regarding process performance. Beyond verification support, modeling tools could easily calculate process metrics to assist the modeler in the design task. Building on such features, the tool vendors could easily provide a greater benefit to their customers and help to improve the process of designing business process models.

4. *Teaching Process Modeling:* Apparently, there is a weakness in the way business process modeling is taught, if practitioners introduce a formal error in every tenth model (at least in our sample). Even worse, the four textbooks on business process modeling that we used to build the holdout sample had an even higher error ratio. While these rates might be partially attributed to missing verification support in the tools, there seems to be a problem for many modelers to understand the behavioral implications of their design. This has two consequences. Firstly, teaching of business process modeling should pay less attention to teaching a specific business process modeling language, but instead focus on conveying the general principles behind it, i.e., concurrency, synchronization, repetition, and other aspects captured, as by the workflow patterns. Secondly, formal errors seem to get too little attention. Concepts like deadlocks should not only be taught as a technical property of a business process model, but also an erroneous business rule that also leads to problems in real-world business processes that are not supported by information systems. Furthermore, the metrics are a good starting point for teaching patterns that are unlikely to result in errors, such as a high degree of structuredness appears

to be less prone to cause errors. Such an approach might eventually deliver a better awareness and attention of formal errors in business process modeling practice.

7.3 Future Research

There are several open questions that could not be addressed in detail in this thesis. In particular, we focused on business process model metrics and their capability to predict errors in business process models. We found strong evidence that our set of metrics can indeed explain a great share of the variation in error probability. Still, there are other factors (see page 175) we did not investigate in detail including personal factors, modeling purpose, domain knowledge, modeling language, or graphical layout that all might be connected with error probability. Furthermore, they might also be related to other important quality aspects that we did not analyze, like maintainability or understandability. In particular, we strongly agree with *Moody* [Moo05] who calls for more empirical research in business process modeling. This thesis and its findings gives an idea of the benefits we might gain from this research, and therefore may be regarded as an encouragement to follow *Moody's* call.

Appendix A

Errors found with *xoEPC*

This appendix shows those EPCs of the SAP Reference Model for which *xoEPC* found errors. The rest size is indicated in brackets. Please note that some models have up to nine problems being identified by *xoEPC*. Those models that are not completely reduced may still include errors that *xoEPC* did not find.

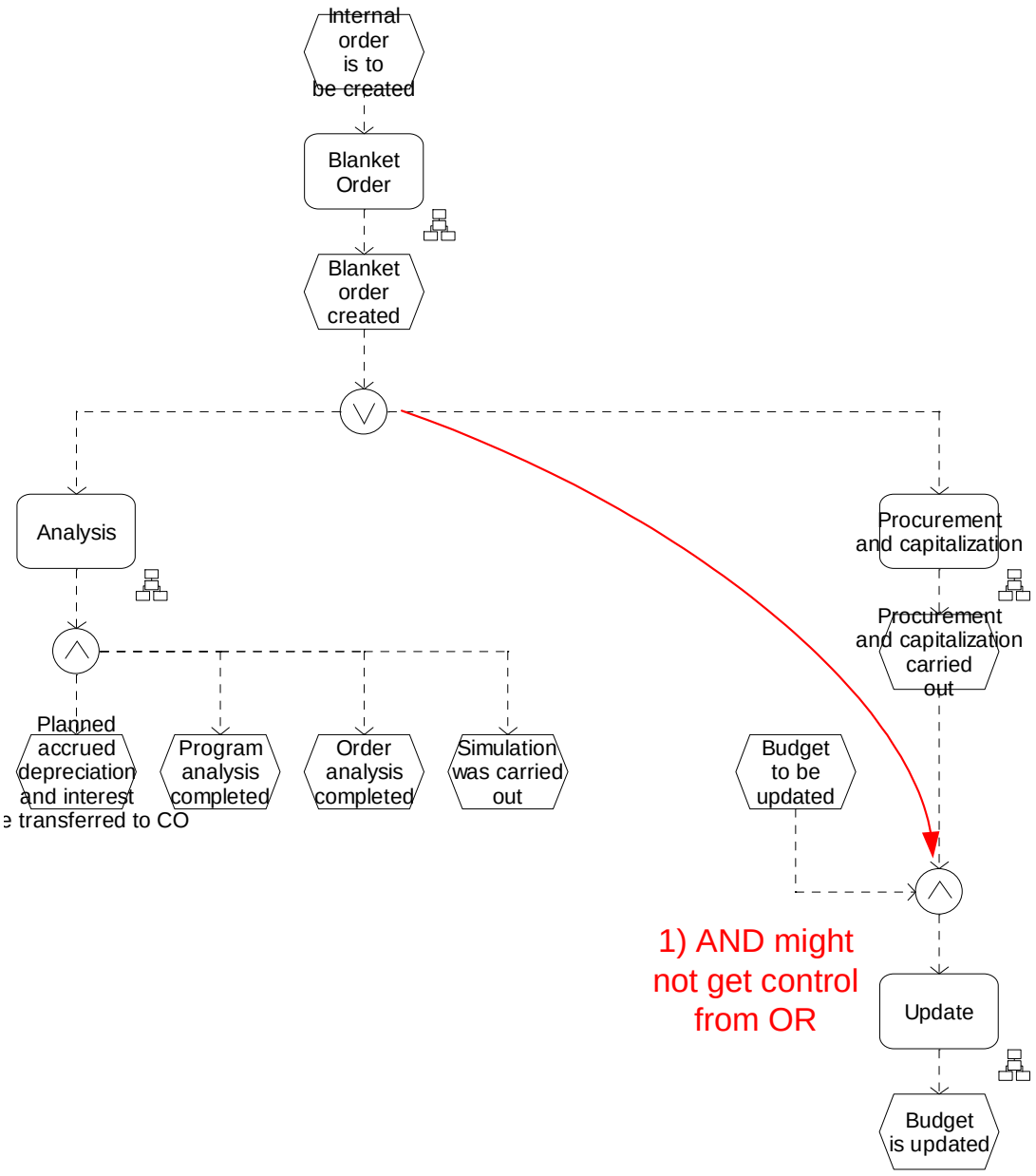


Figure A.1: Asset Accounting – Direct Capitalization (completely reduced)

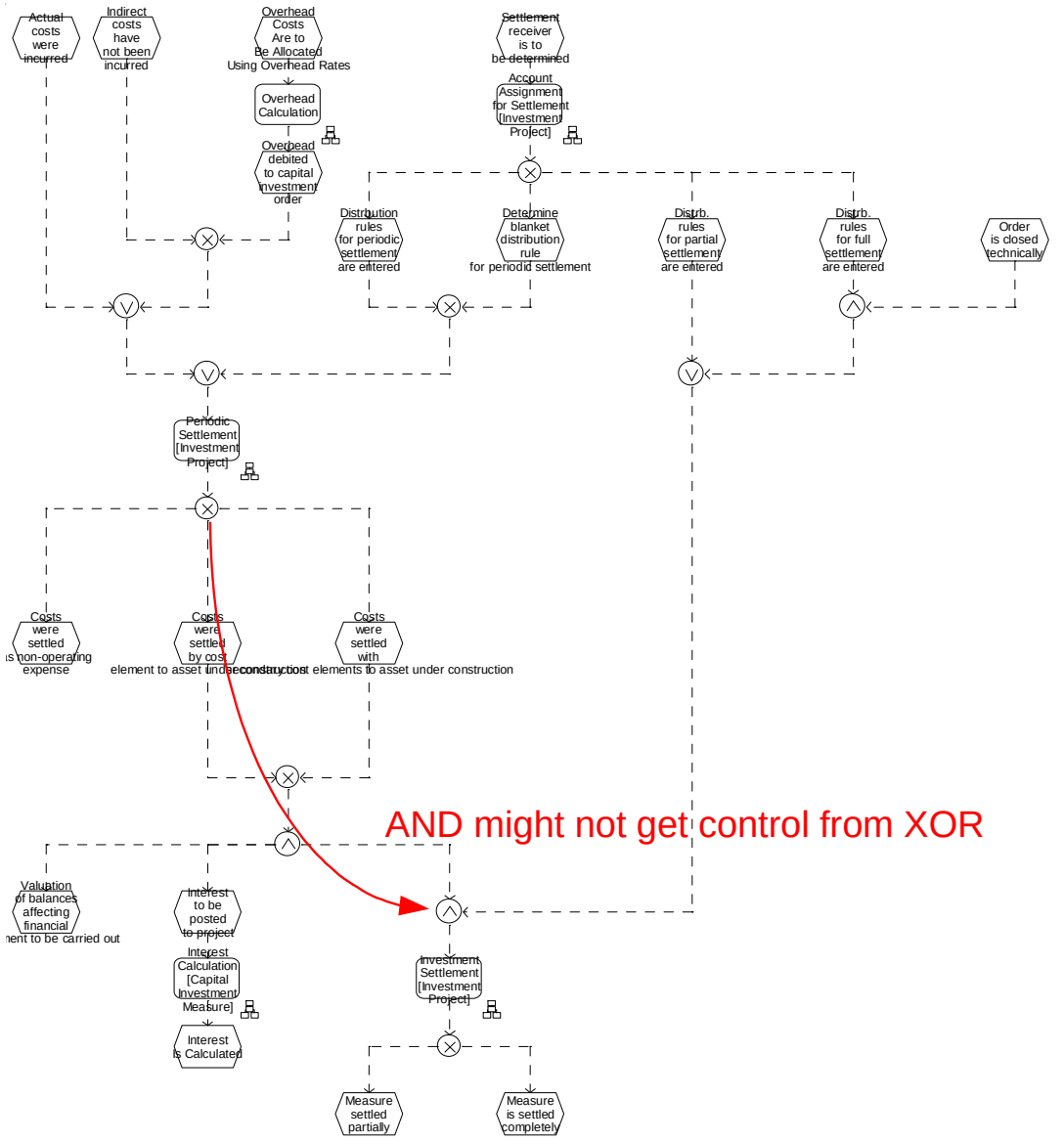


Figure A.2: Asset Accounting – Handling Complex Investment Measures – Period-End Closing and Settlement (reduced size 9)

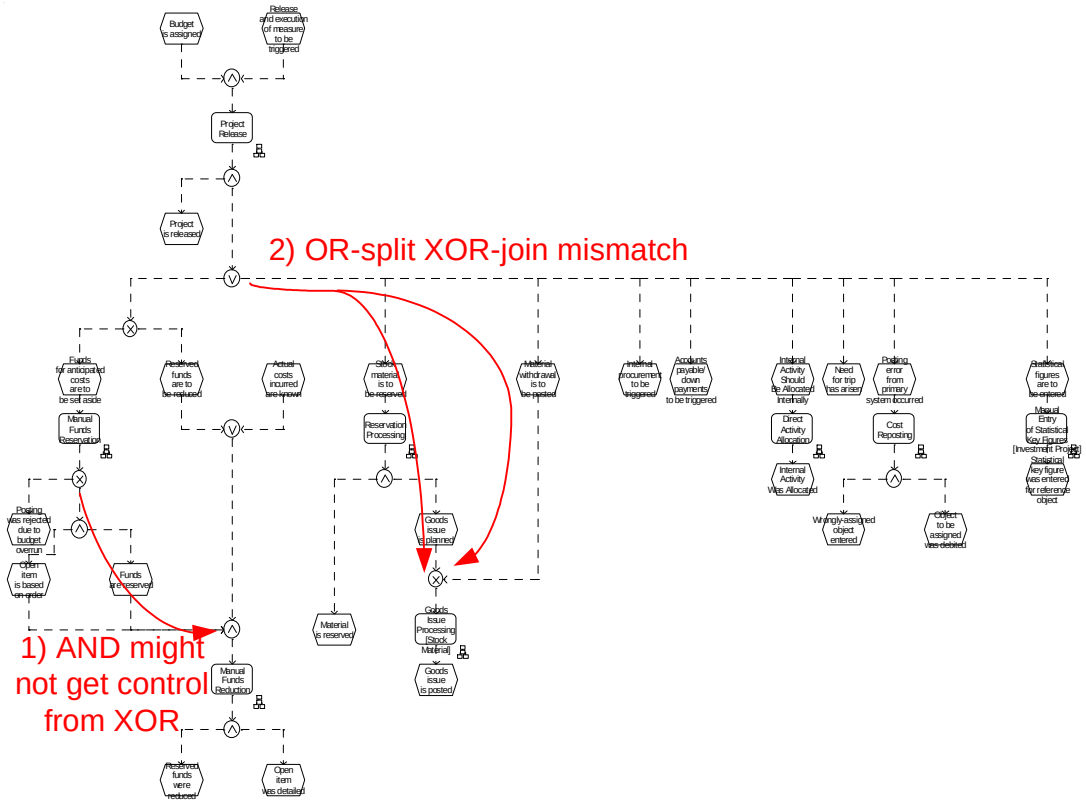


Figure A.3: Asset Accounting – Handling Complex Investment Measures – Release and Implementation of Measure (reduced size 8)

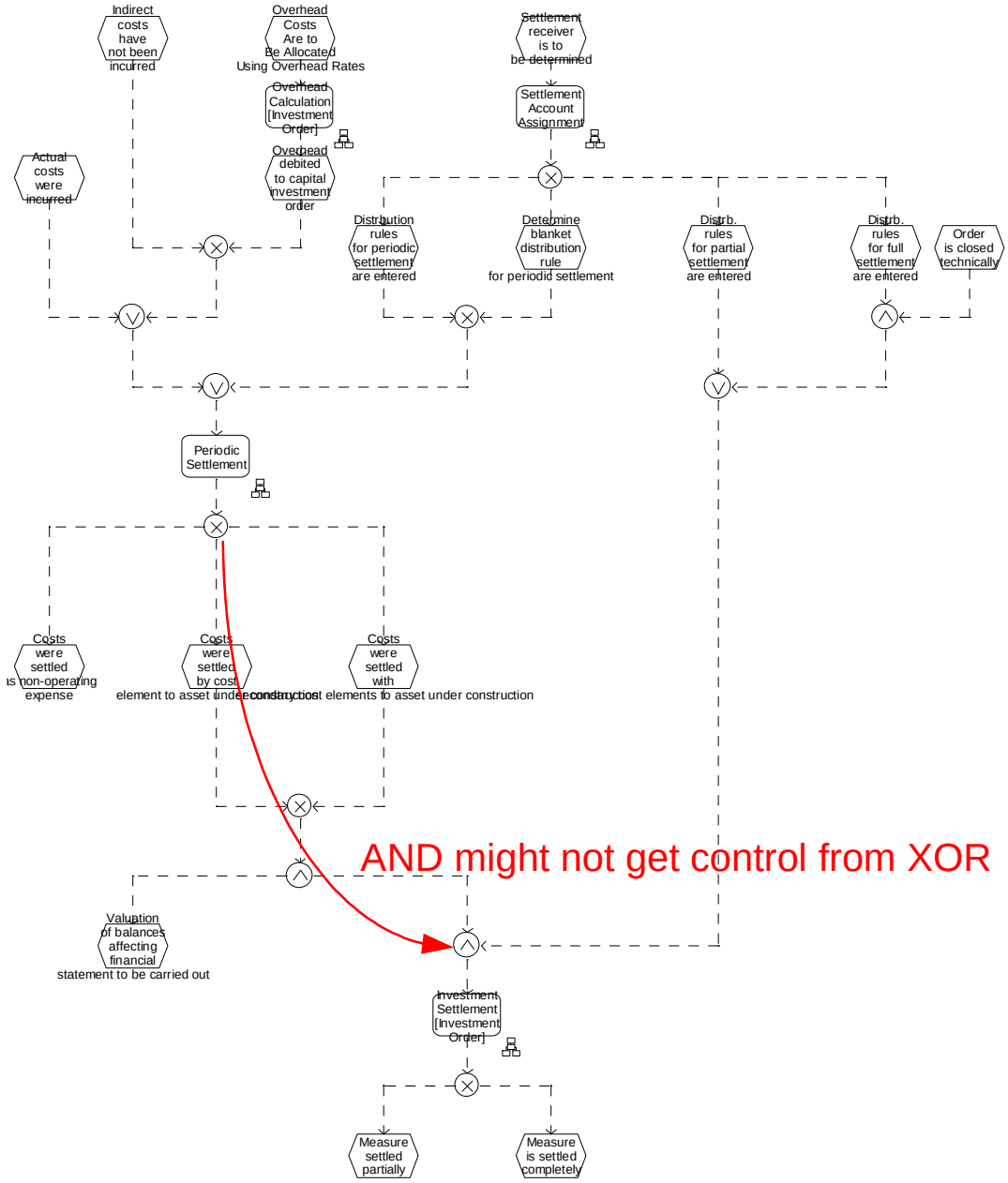


Figure A.4: Asset Accounting – Handling Simple Investment Measures – Period-End Closing and Settlement (reduced size 9)

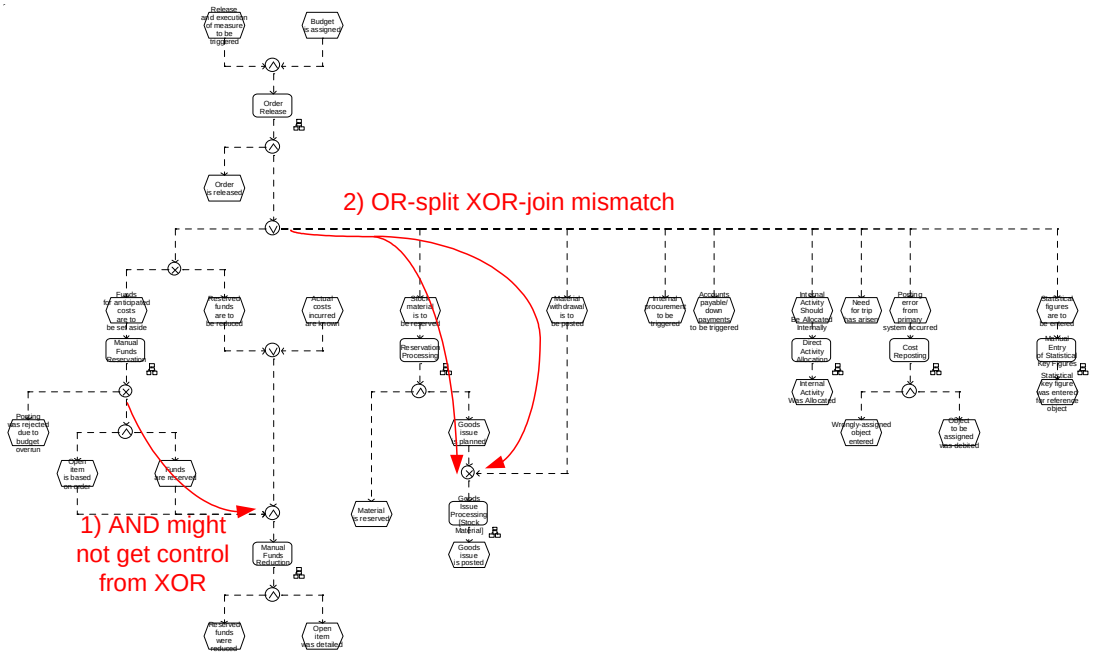


Figure A.5: Asset Accounting – Handling Simple Investment Measures – Release and Implementation of Measure (reduced size 8)

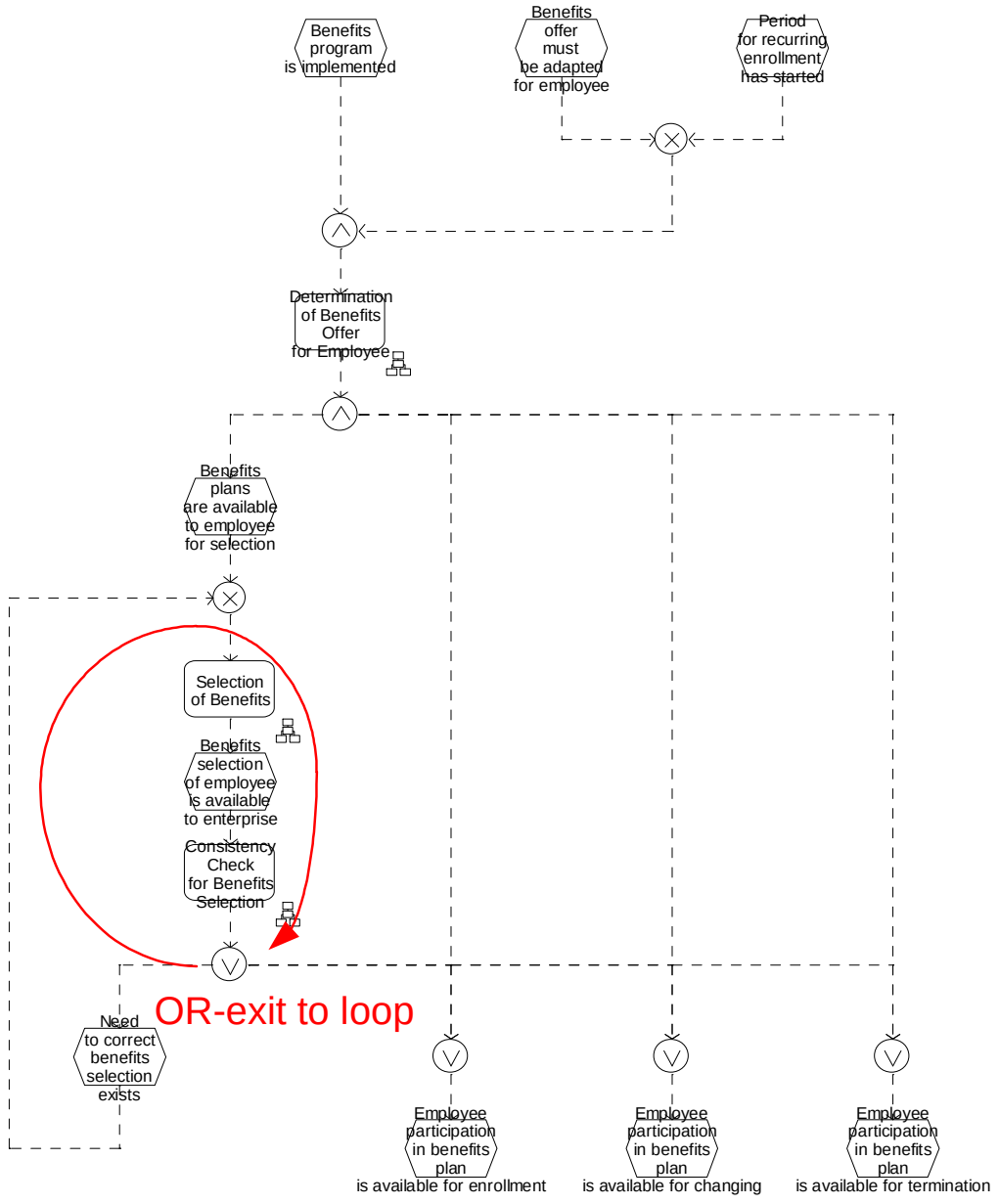


Figure A.6: Benefits Administration – Benefits Administration – Benefits Selection (reduced size 9)

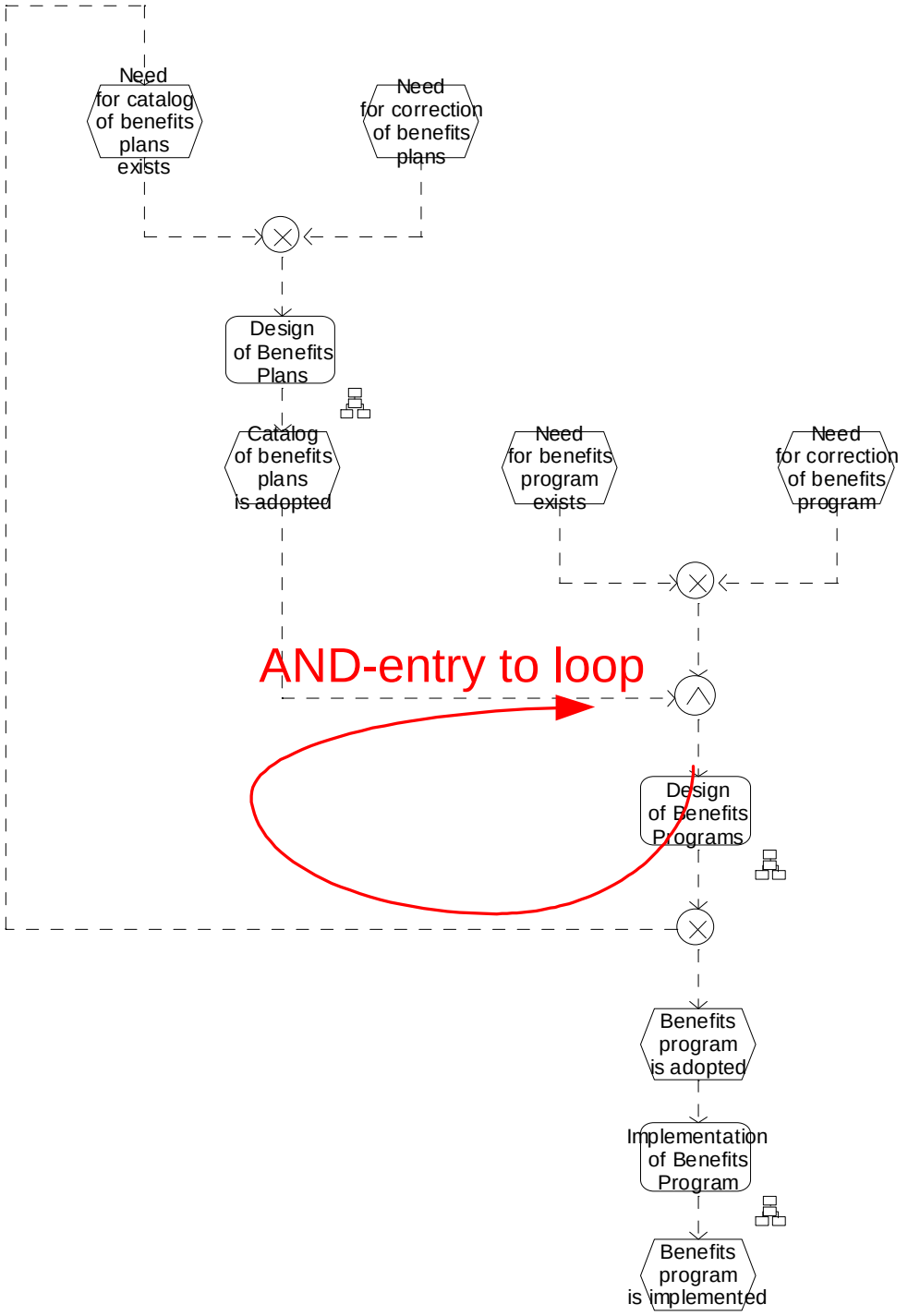


Figure A.7: Benefits Administration – Benefits Administration – Design of Enterprise Benefits System (completely reduced)

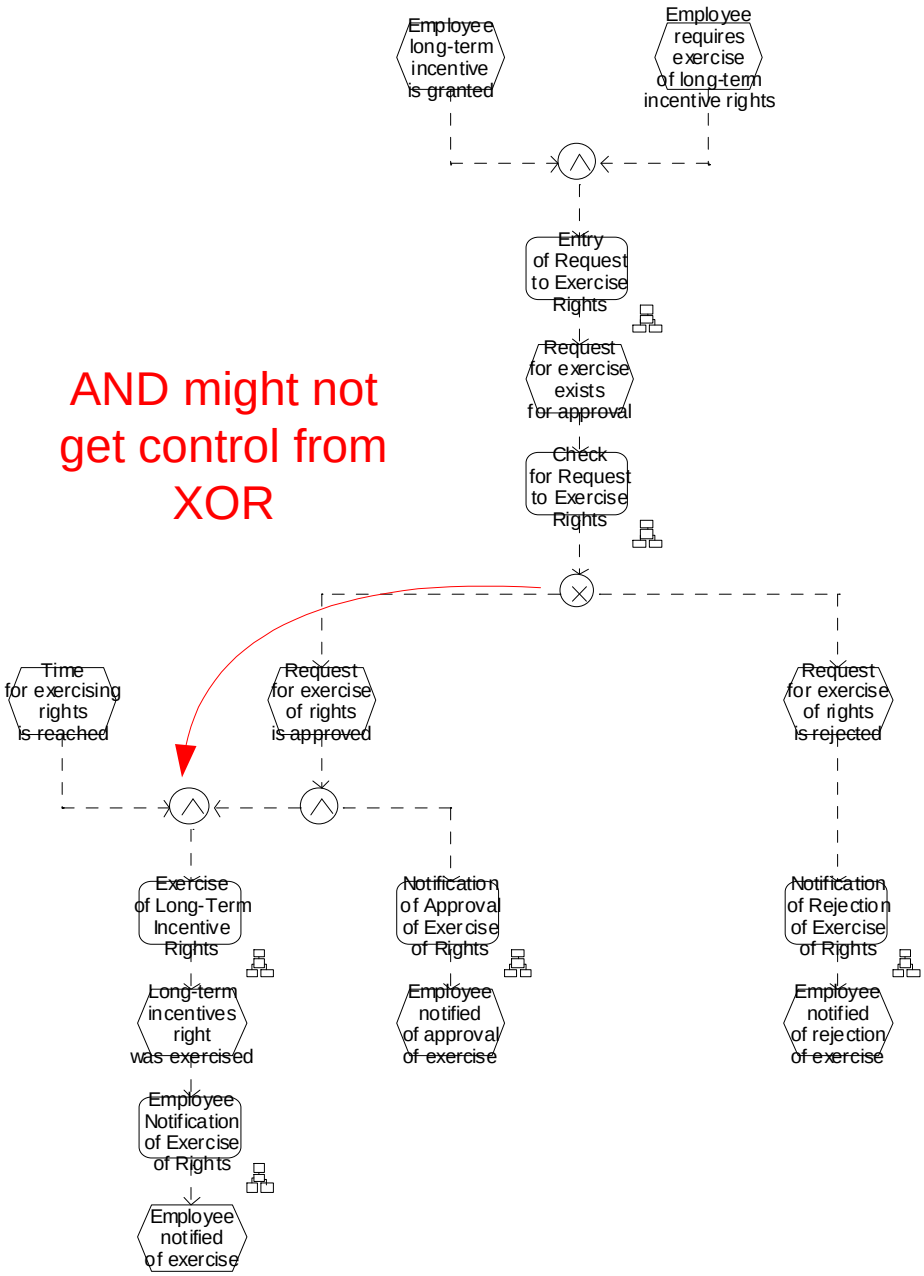


Figure A.8: Compensation Management – Long-Term Incentives – Exercise of Long-Term Incentive Rights by Employee (completely reduced)

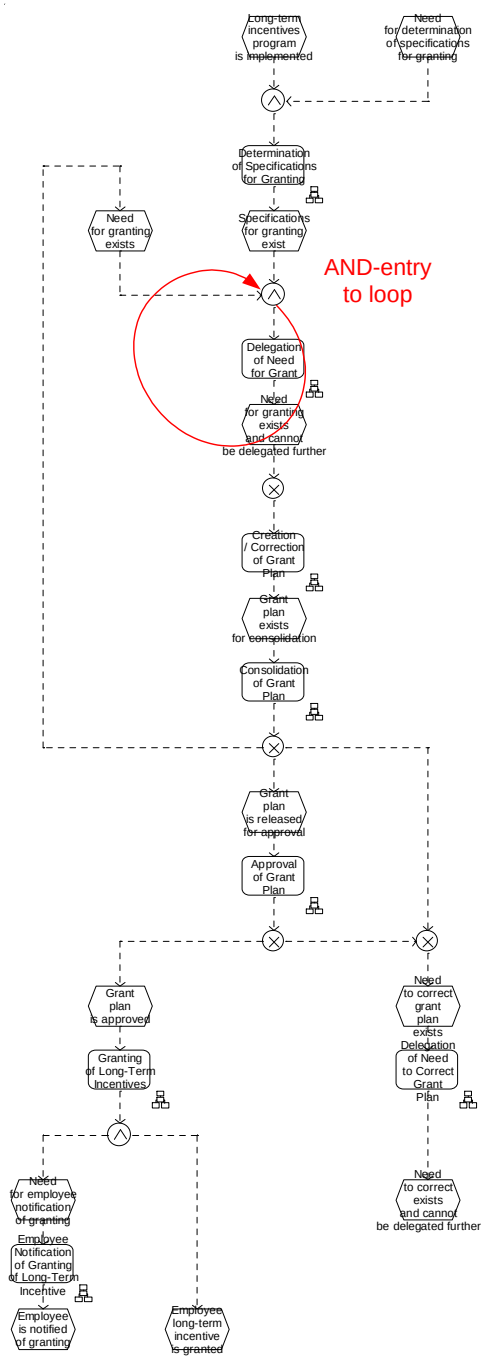


Figure A.9: Compensation Management – Long-Term Incentives – Granting of Share of Long-Term Incentive to Employee (completely reduced)

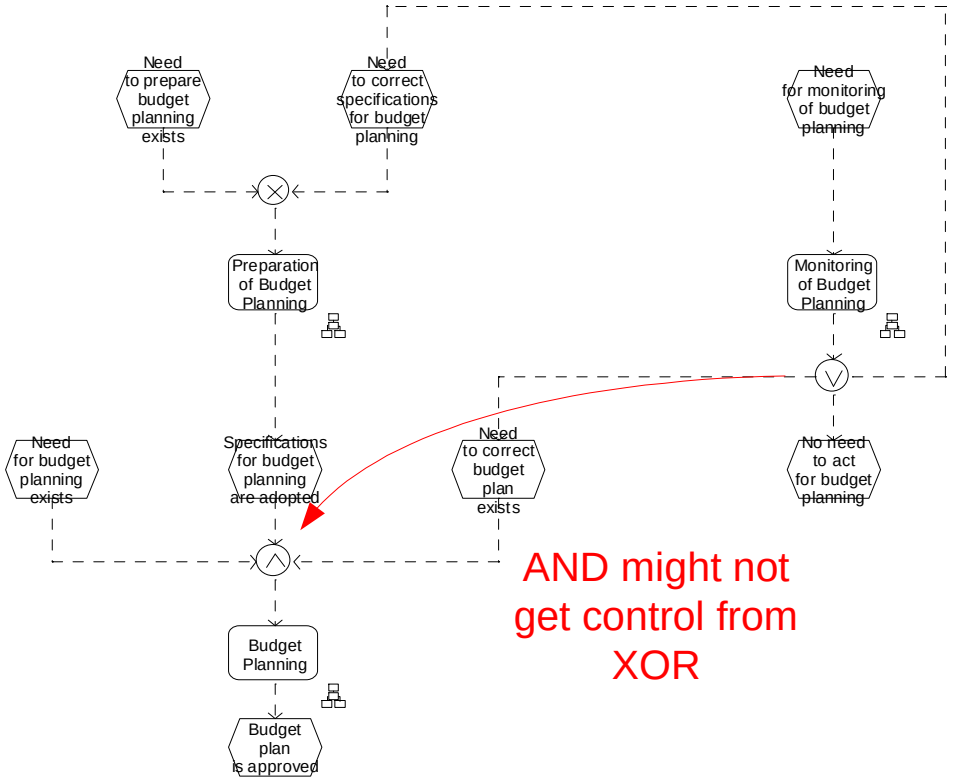


Figure A.10: Compensation Management – Personnel Budget Planning (reduced size 7)

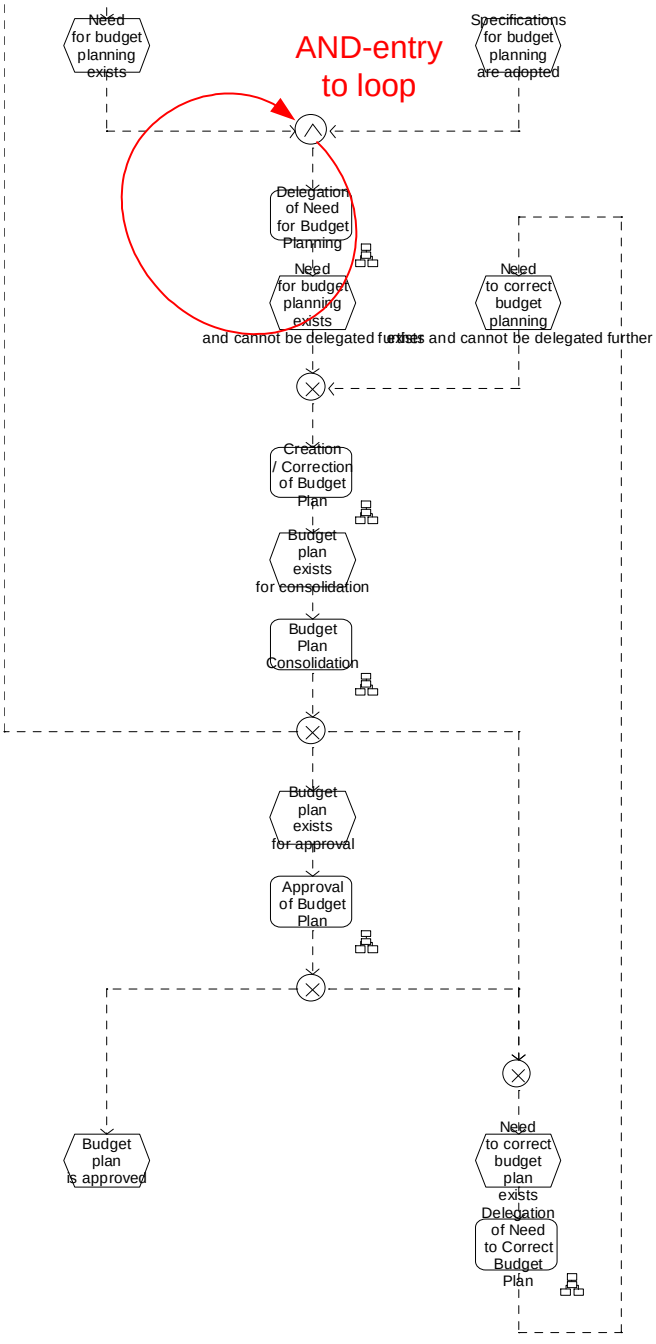


Figure A.11: Compensation Management – Personnel Budget Planning – Budget Planning (completely reduced)

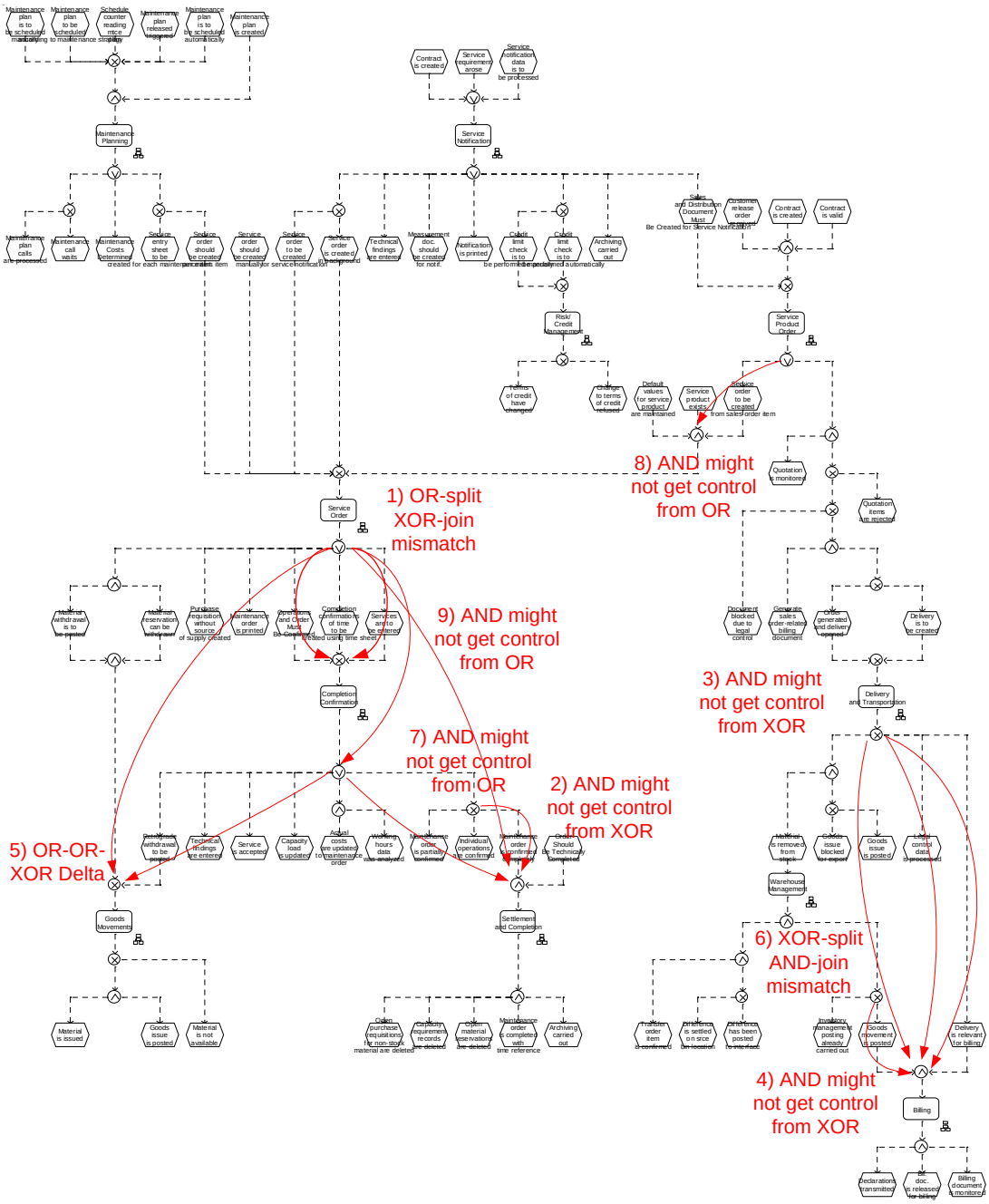


Figure A.12: Customer Service – Repairs Processing at Customer (Field Service) (reduced size 17)

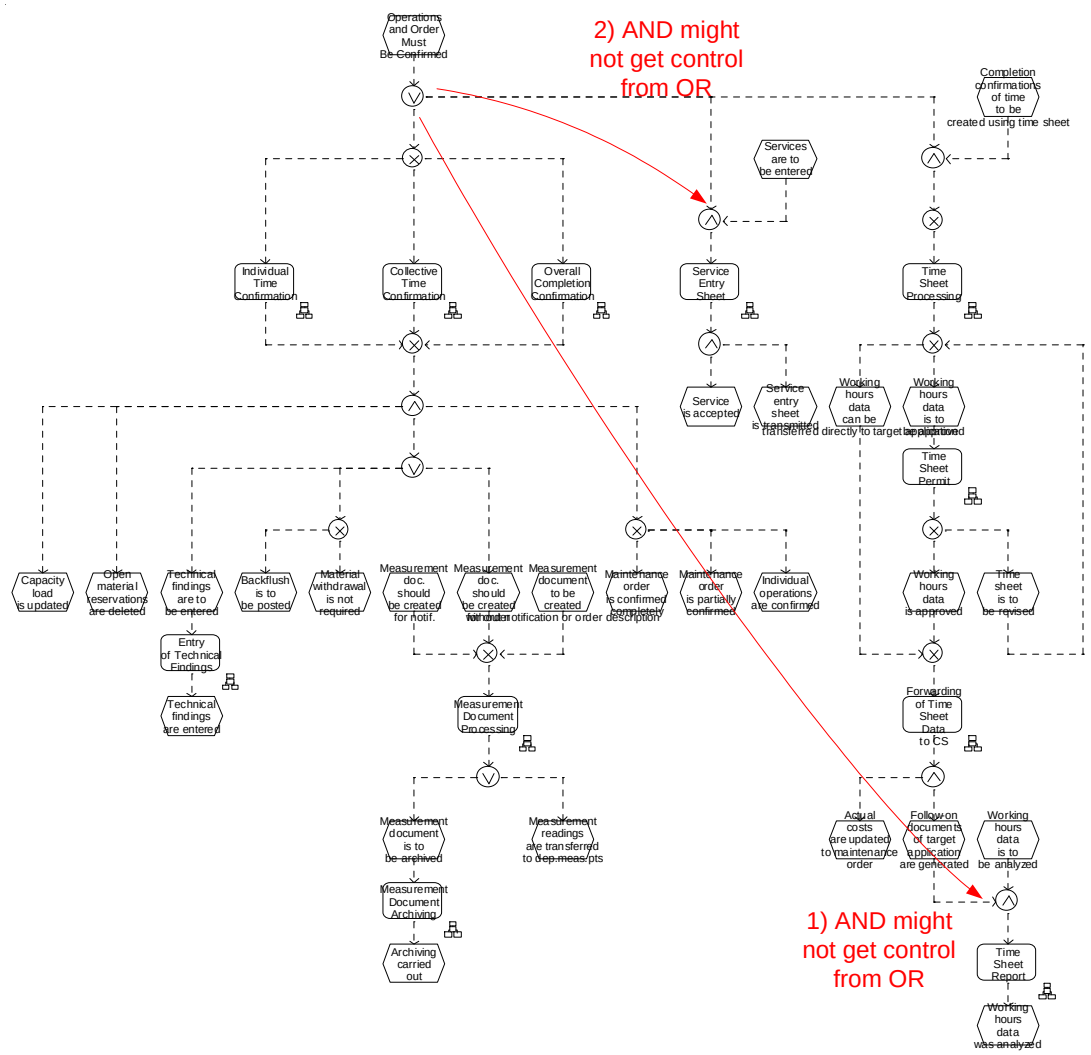


Figure A.13: Customer Service – Repairs Processing at Customer (Field Service) – Completion Confirmation (reduced size 11)

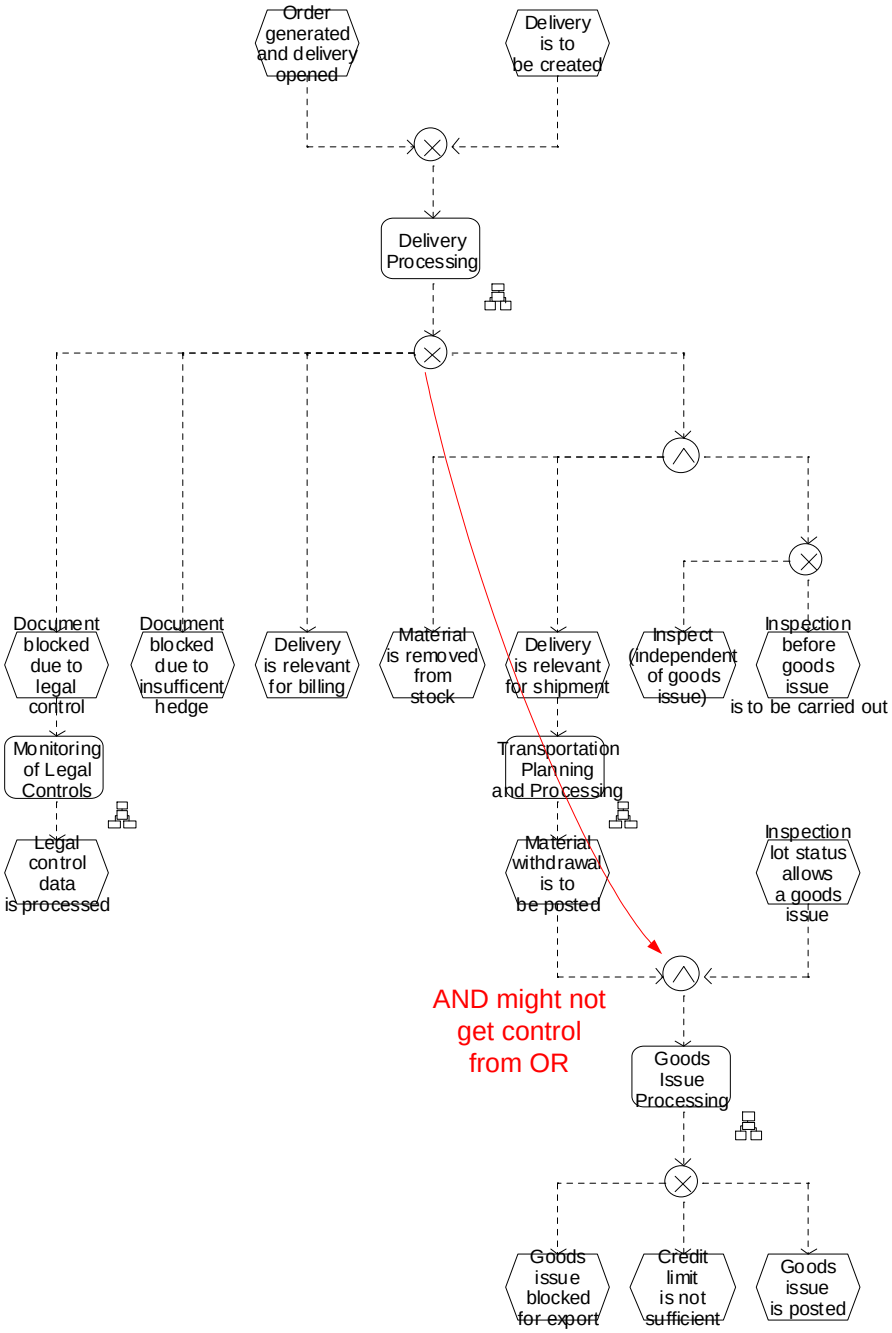


Figure A.14: Customer Service – Repairs Processing at Customer (Field Service) – Delivery and Transportation (completely reduced)

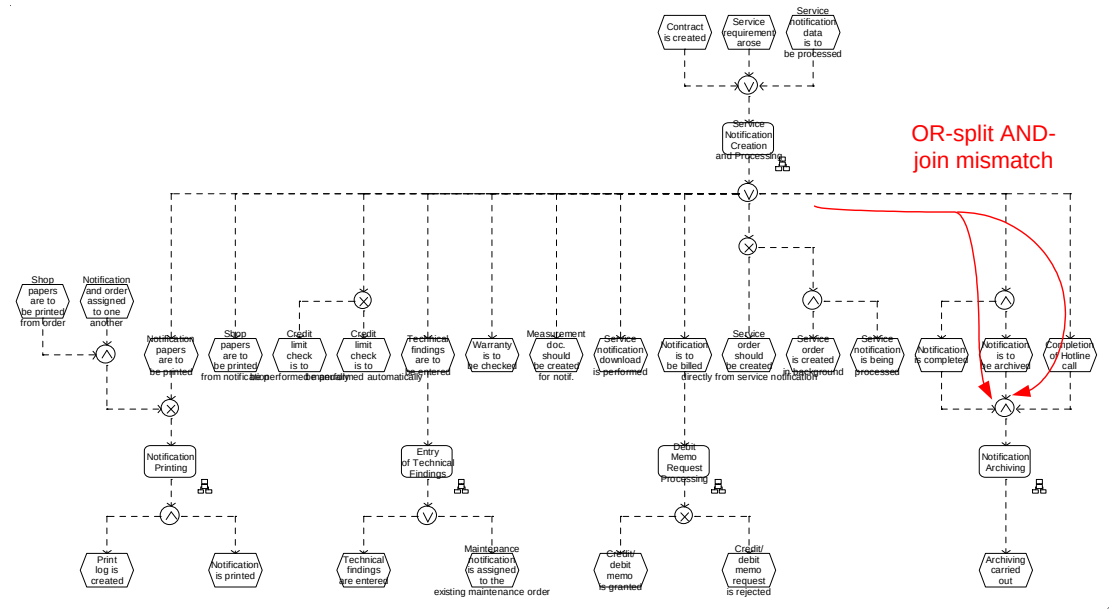


Figure A.15: Customer Service – Repairs Processing at Customer (Field Service) – Service Notification (reduced size 6)

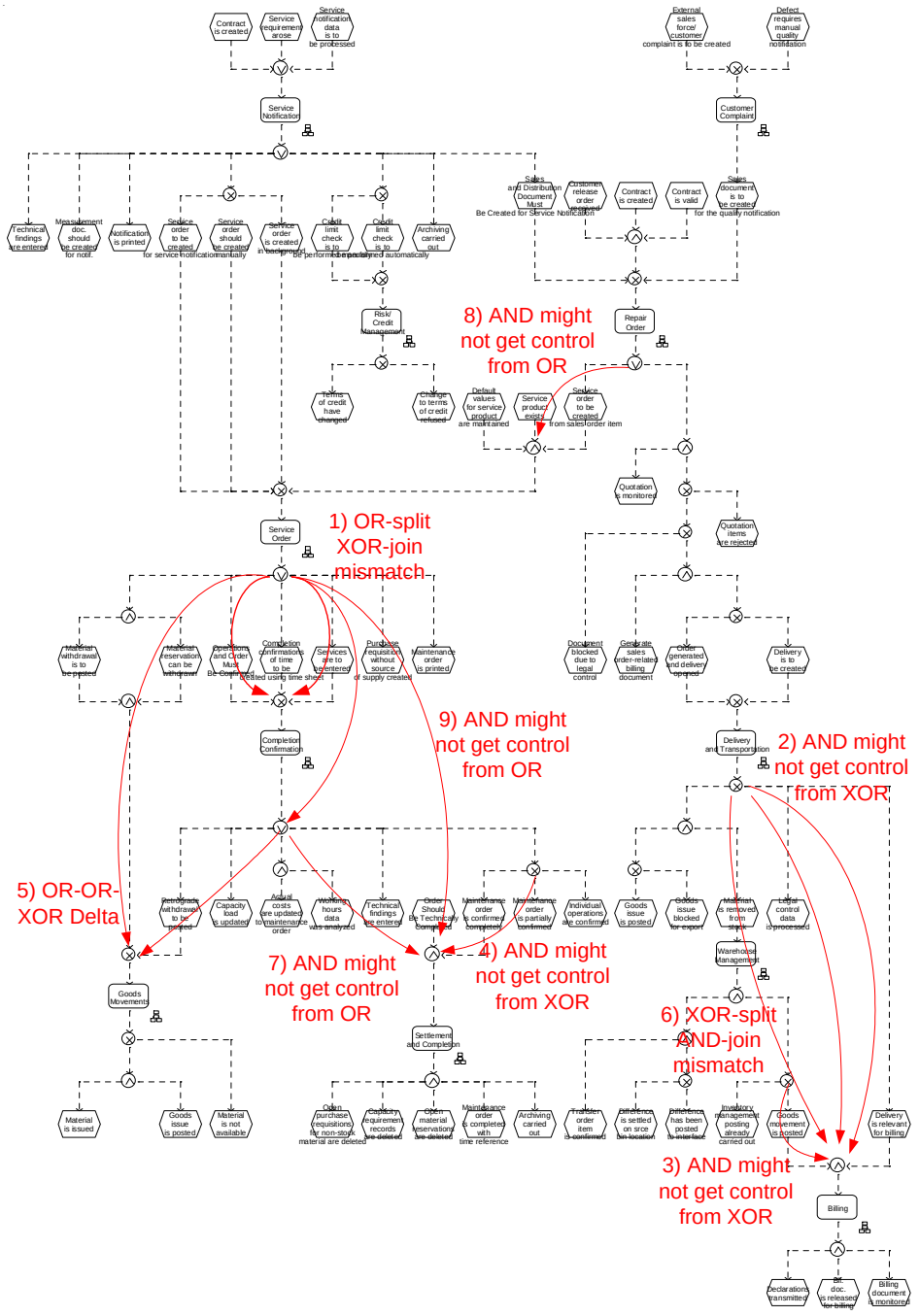


Figure A.16: Customer Service – Repairs Processing in Service Center (Inhouse) (reduced size 12)

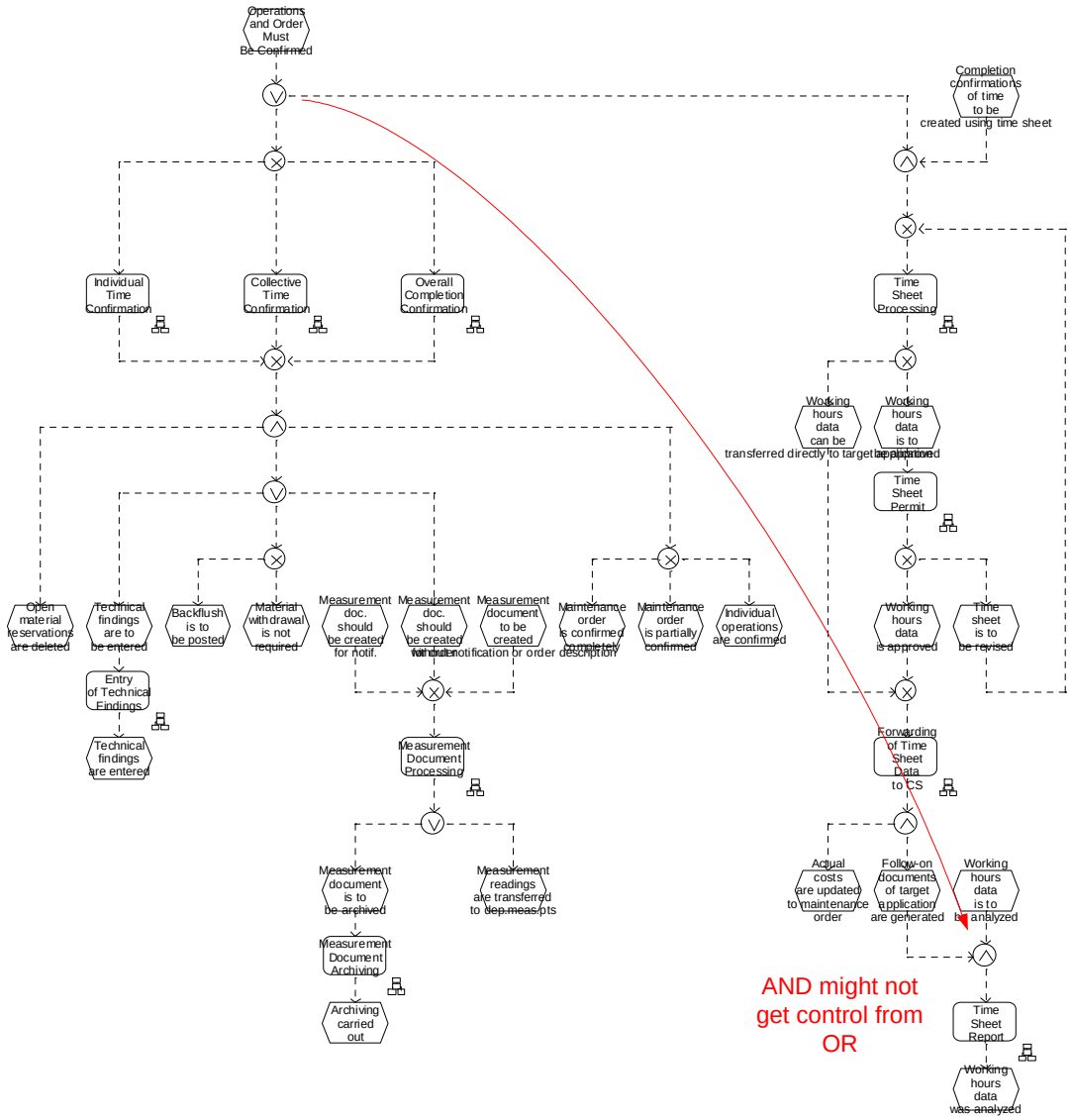


Figure A.17: Customer Service – Repairs Processing in Service Center (Inhouse) – Completion Confirmation (reduced size 8)

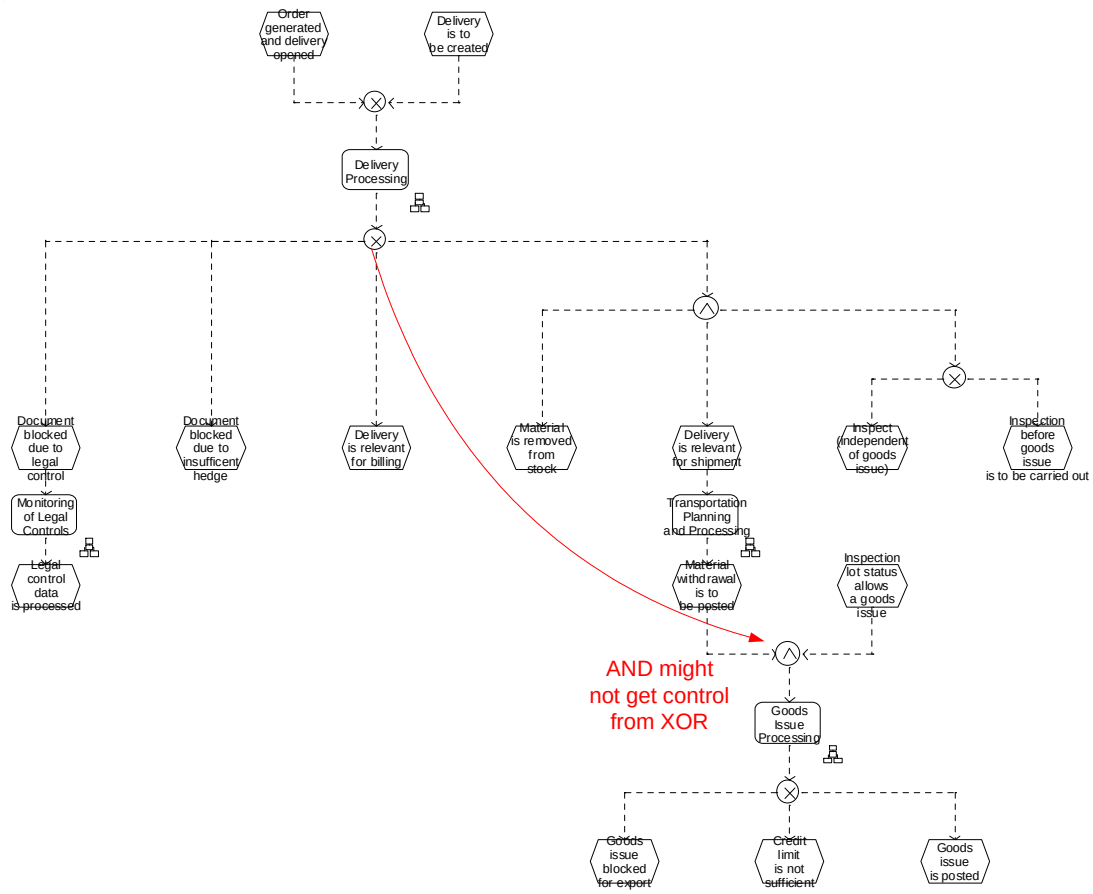


Figure A.18: Customer Service – Spare Parts Delivery Processing – Delivery and Transportation (completely reduced)

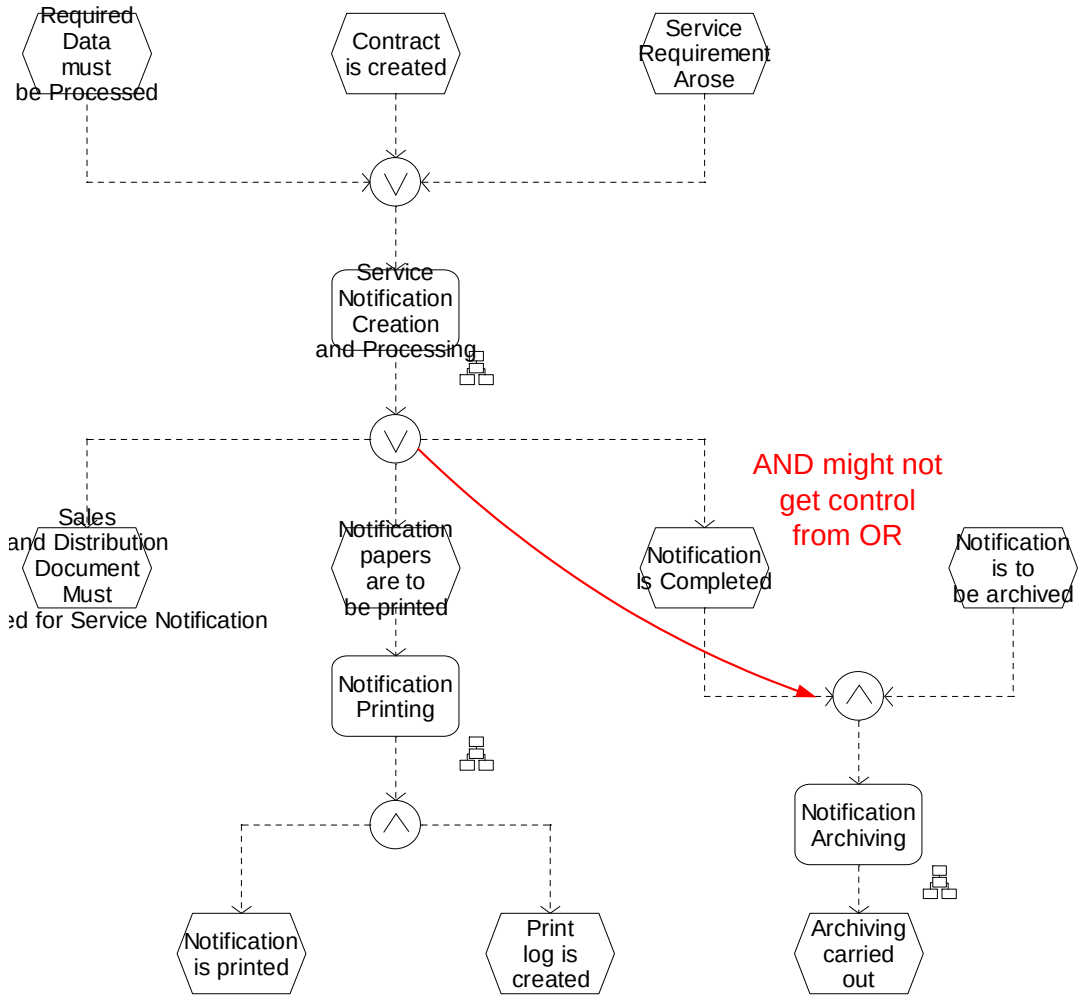


Figure A.19: Customer Service – Spare Parts Delivery Processing – Service Notification (completely reduced)

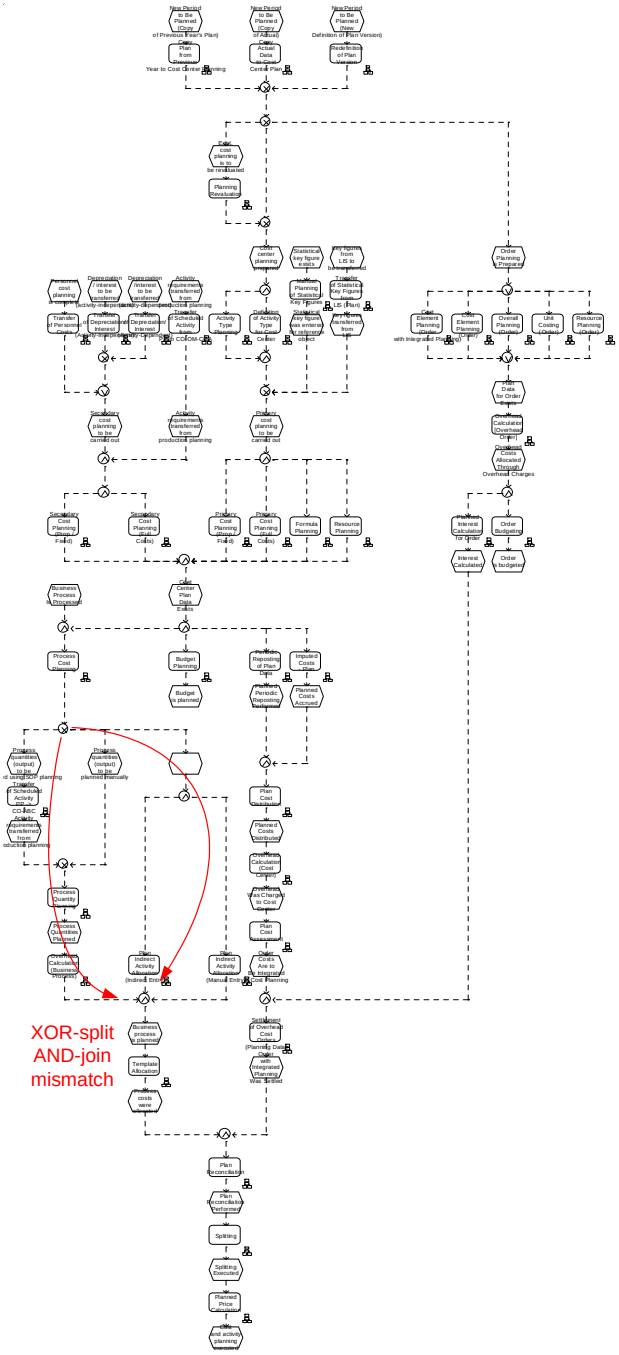


Figure A.20: Enterprise Controlling – Operational business planning – Cost and Activity Planning (reduced size 7)

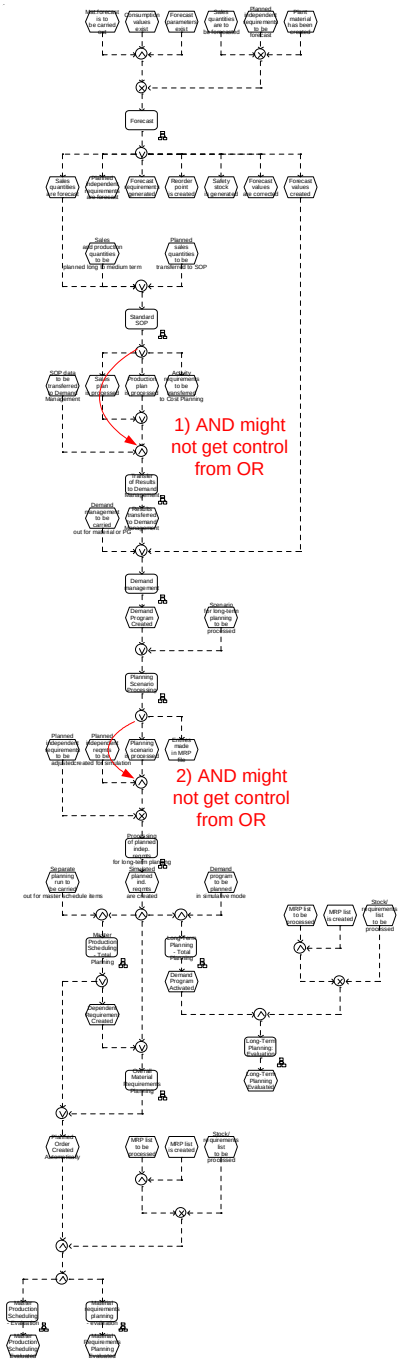


Figure A.21: Enterprise Controlling – Operational business planning – Production Planning (reduced size 23)

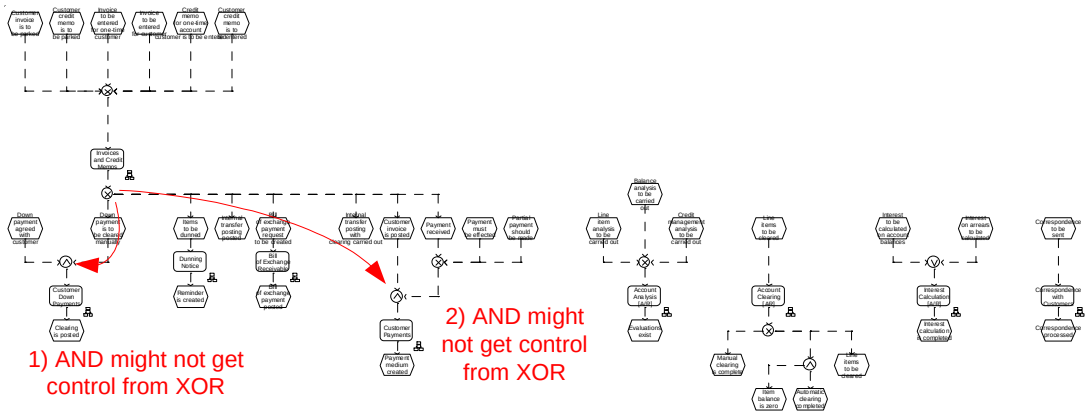


Figure A.22: Financial Accounting – Accounts Receivable (reduced size 9)

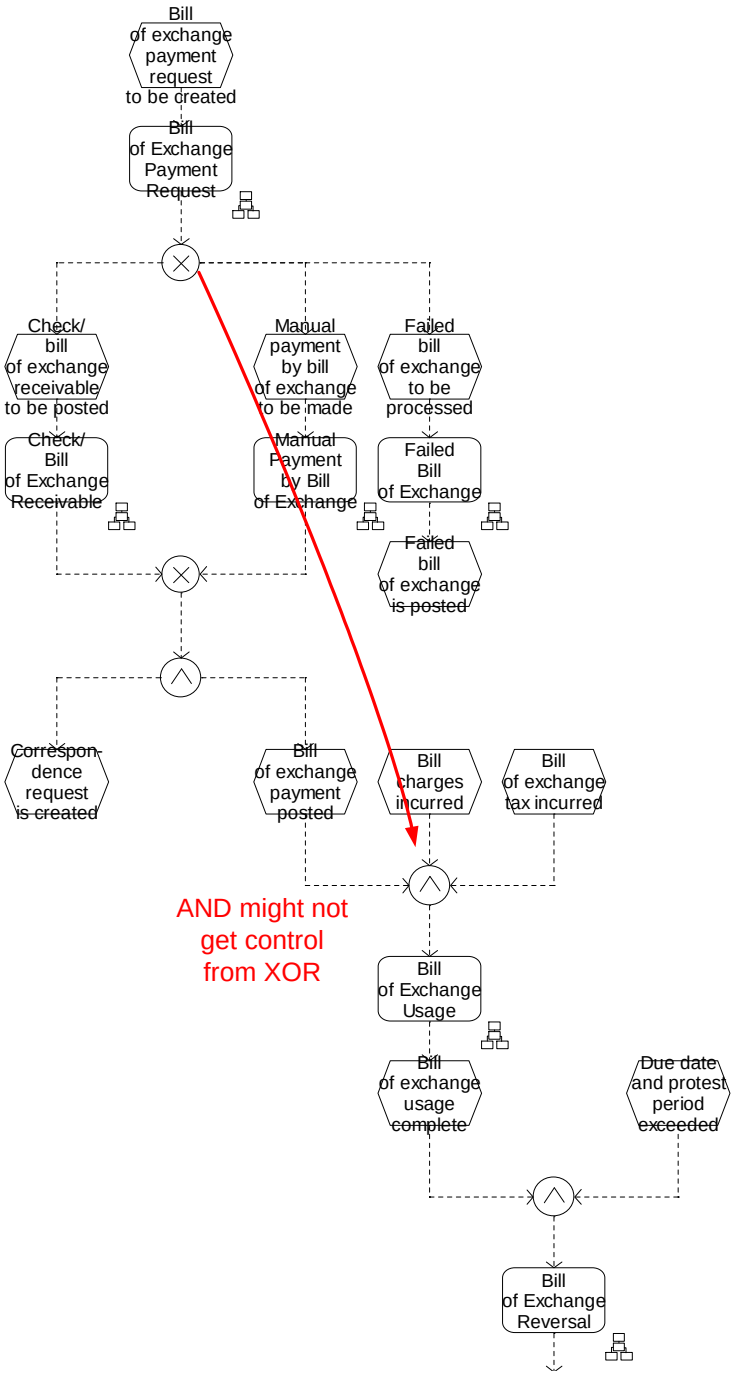


Figure A.23: Financial Accounting – Accounts Receivable – Bill of Exchange Receivable (completely reduced)

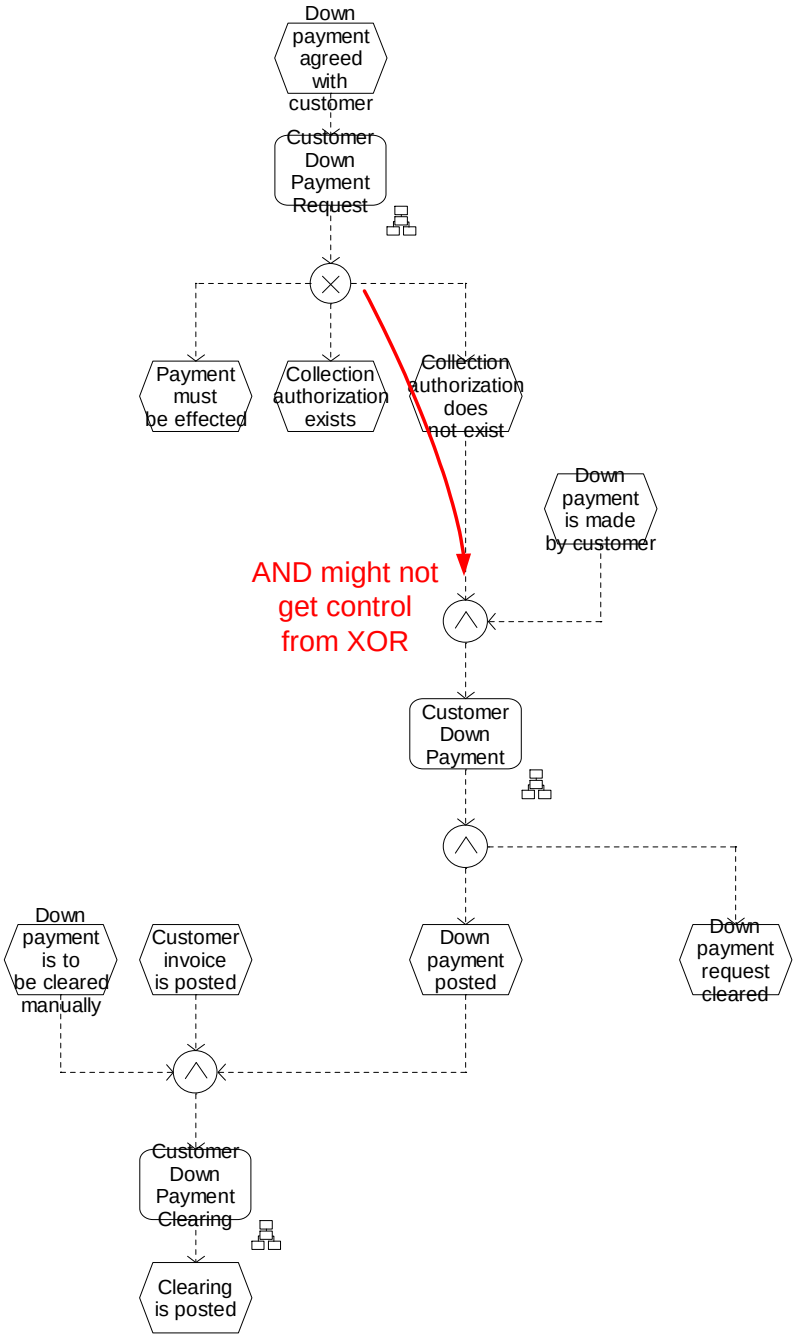


Figure A.24: Financial Accounting – Accounts Receivable – Customer Down Payments (completely reduced)

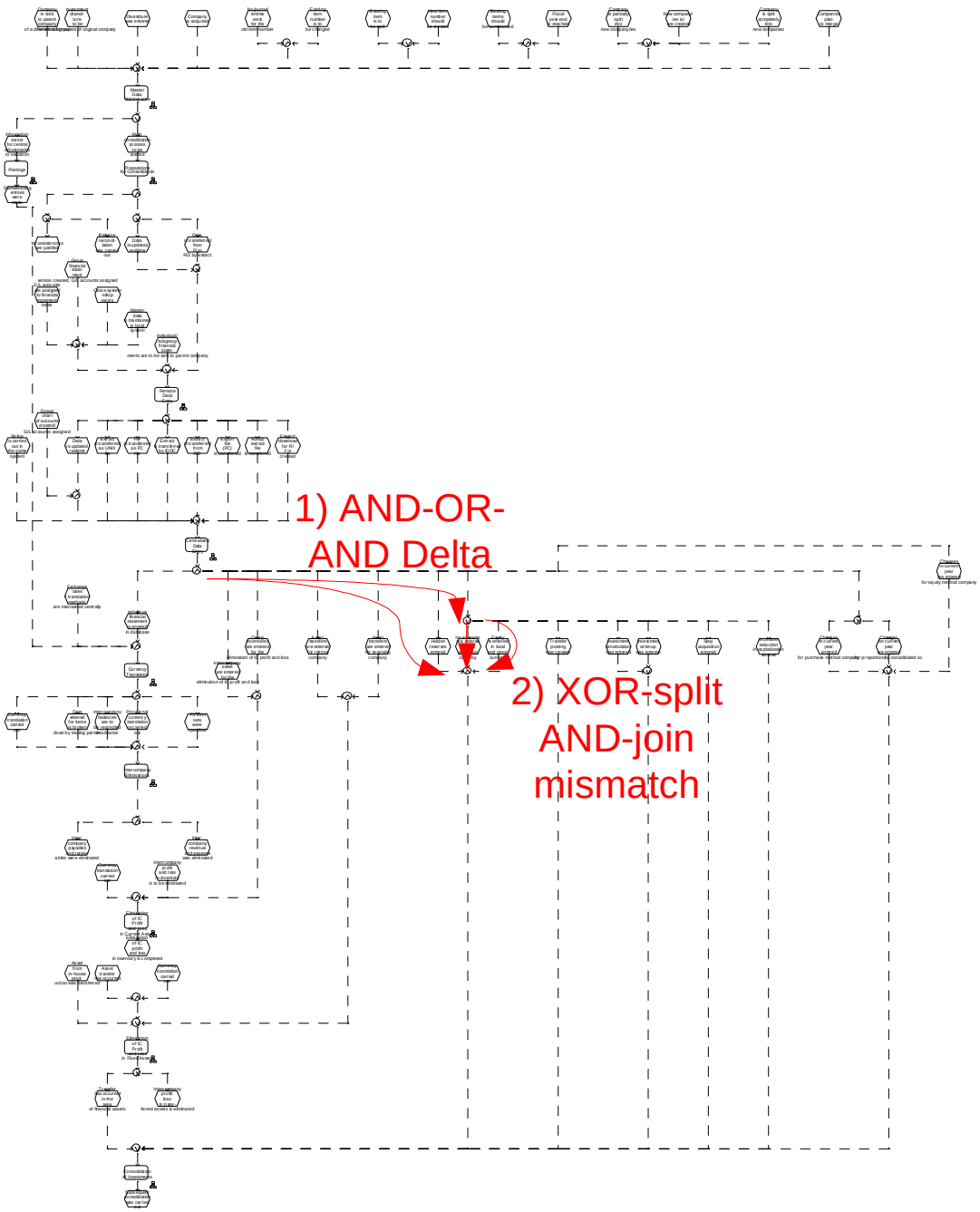


Figure A.25: Financial Accounting – Consolidation (reduced size 22)

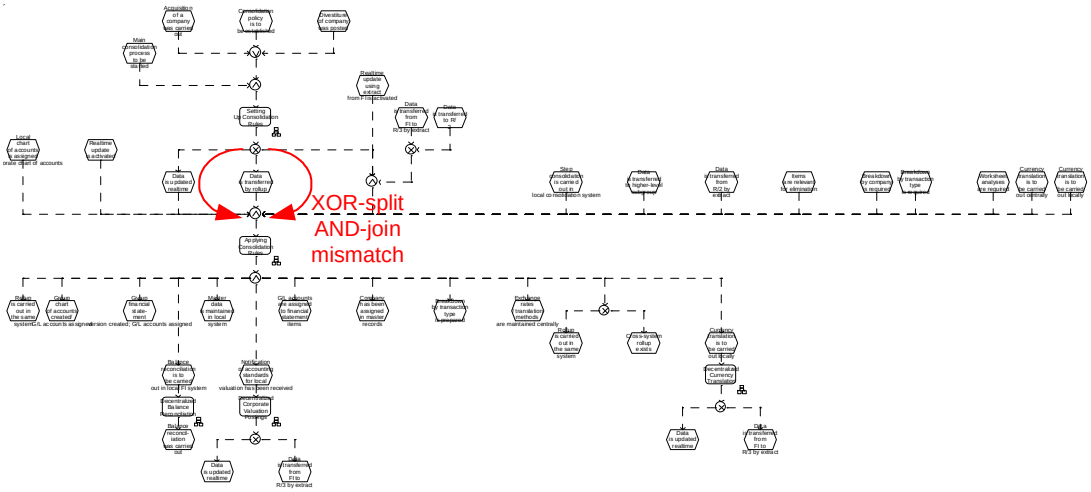


Figure A.26: Financial Accounting – Consolidation – Preparations for Consolidation (completely reduced)

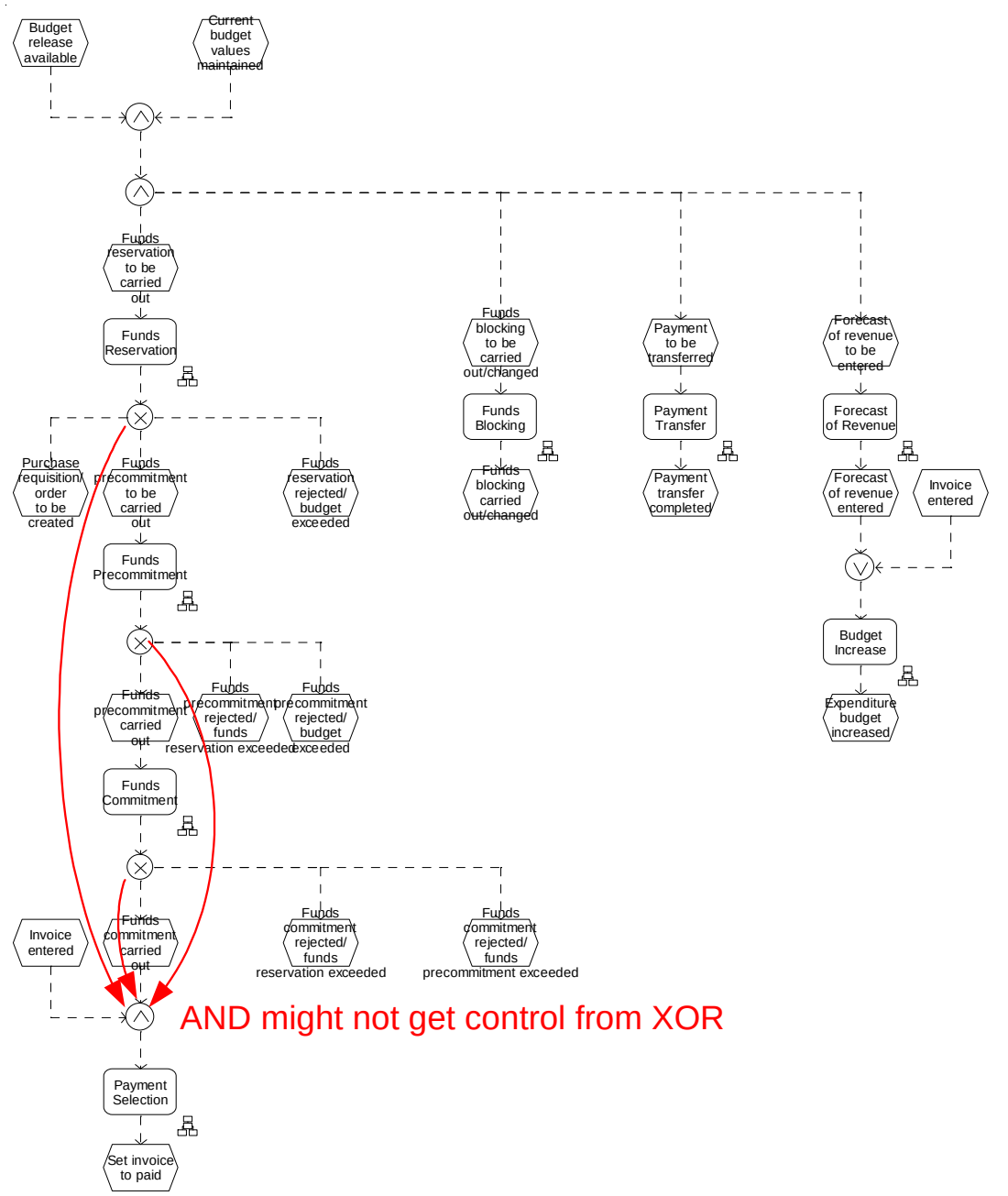


Figure A.27: Financial Accounting – Funds Management – Budget Execution (completely reduced)

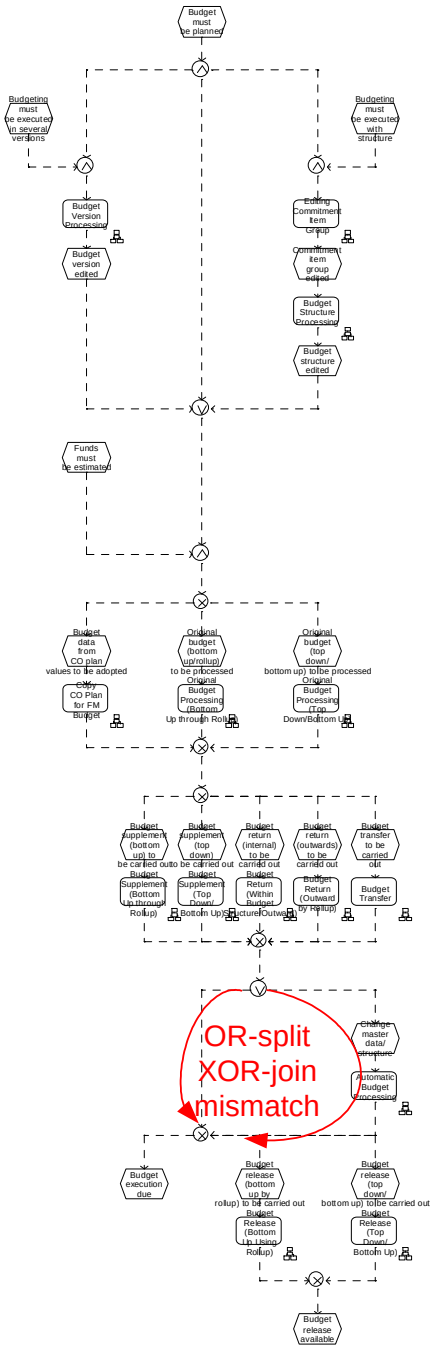


Figure A.28: Financial Accounting – Funds Management – Budget Planning (completely reduced)

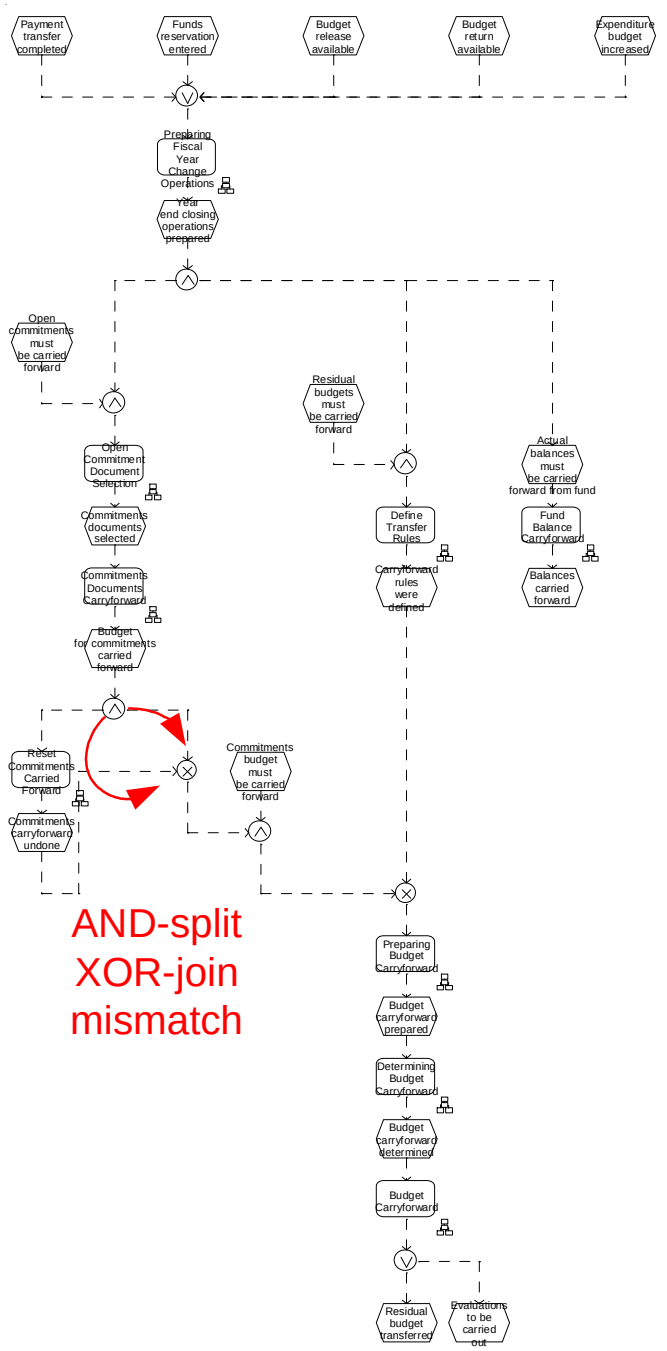


Figure A.29: Financial Accounting – Funds Management – Fiscal Year Change Operations (Funds Management) (reduced size 8)

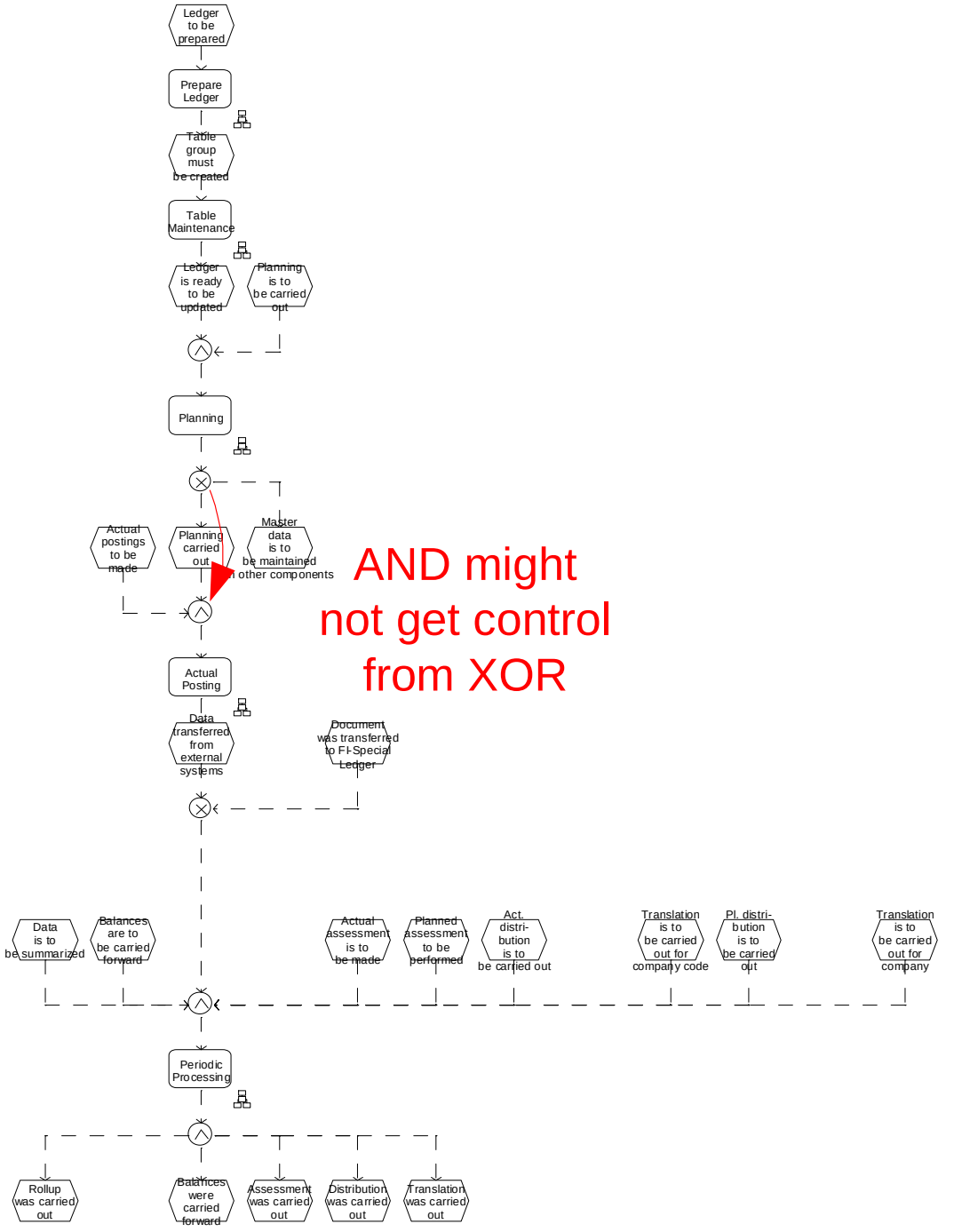


Figure A.30: Financial Accounting – Special Purpose Ledger (completely reduced)

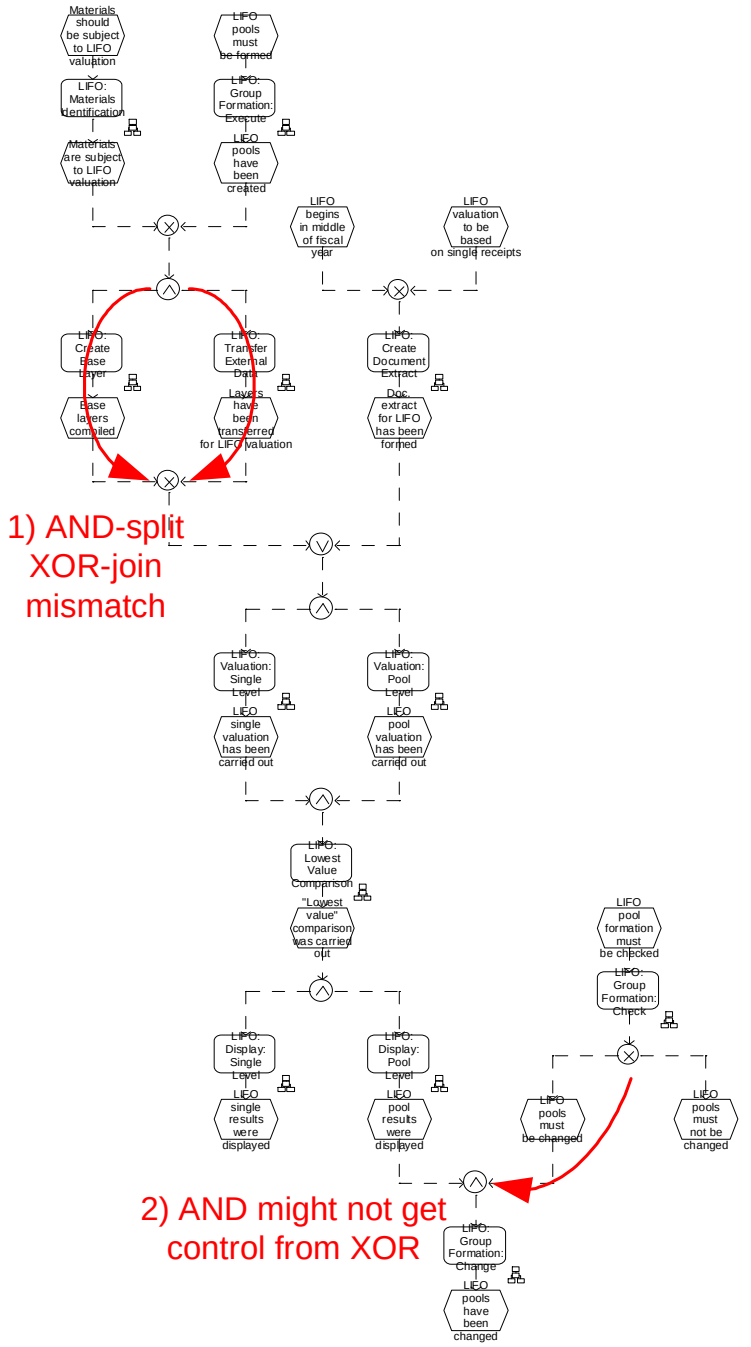


Figure A.31: Financial Accounting – Valuation of Balances Relevant to Balance Sheet – LIFO valuation (completely reduced)

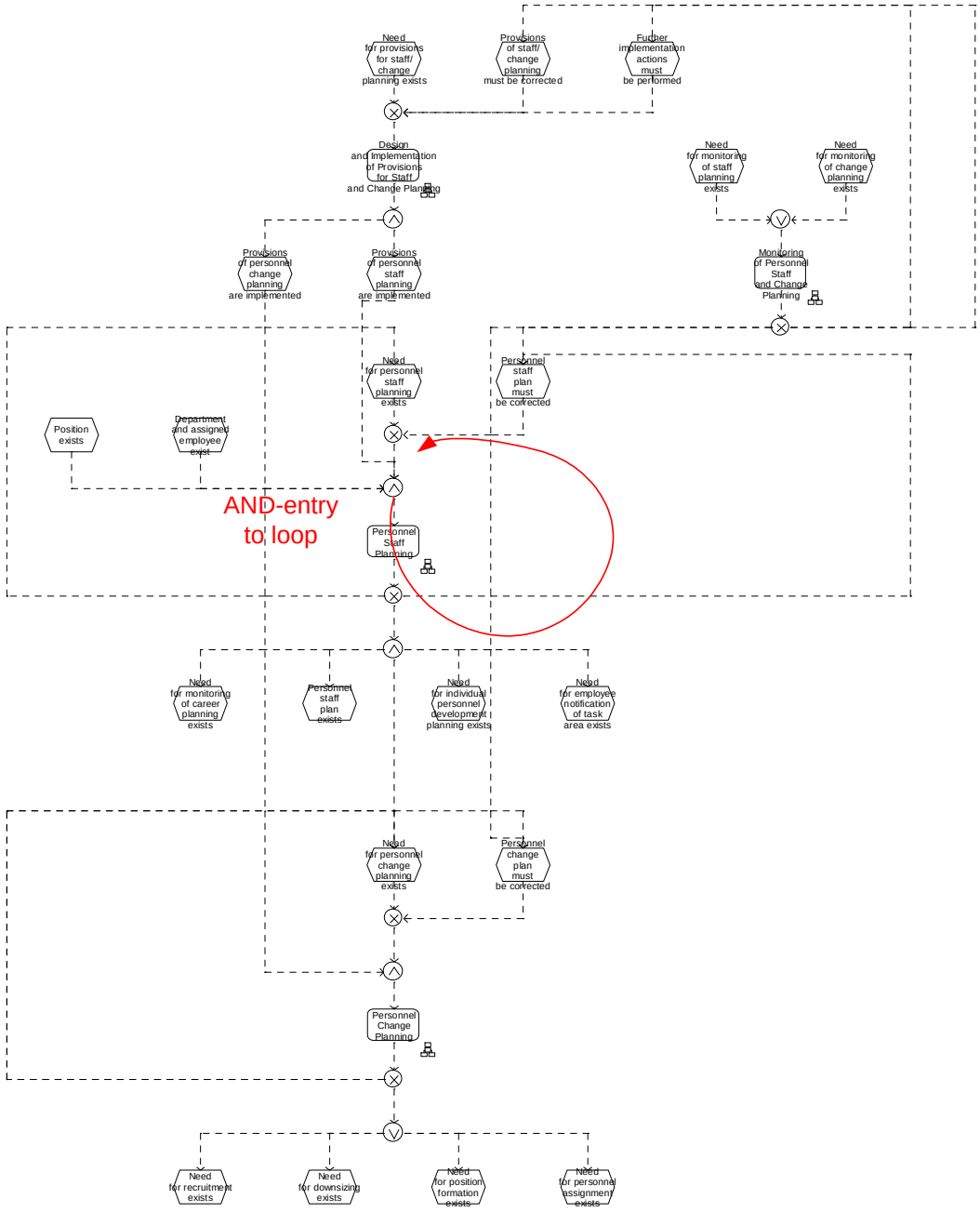


Figure A.32: Organizational Management – Planning Staff Assignment and Changes (reduced size 15)

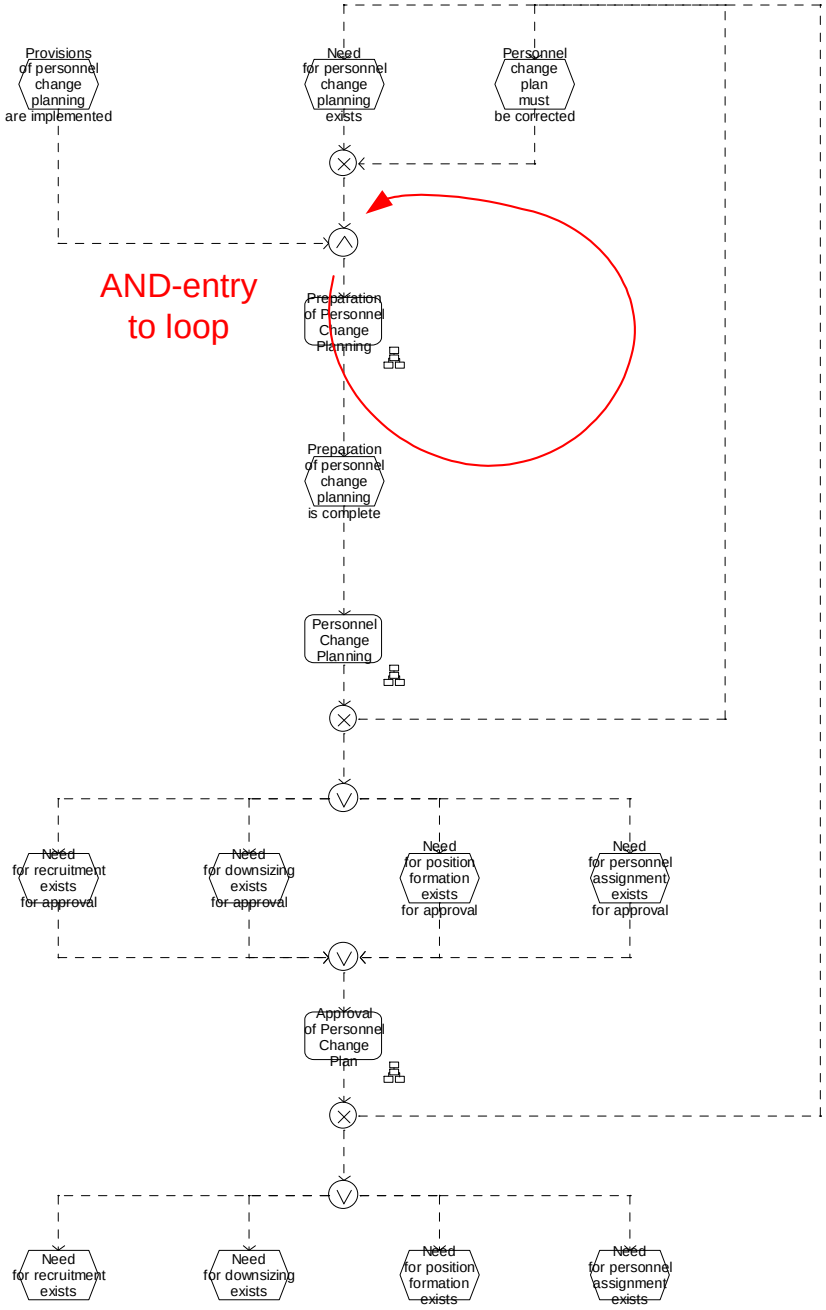


Figure A.33: Organizational Management – Planning Staff Assignment and Changes – Personnel Change Planning (completely reduced)

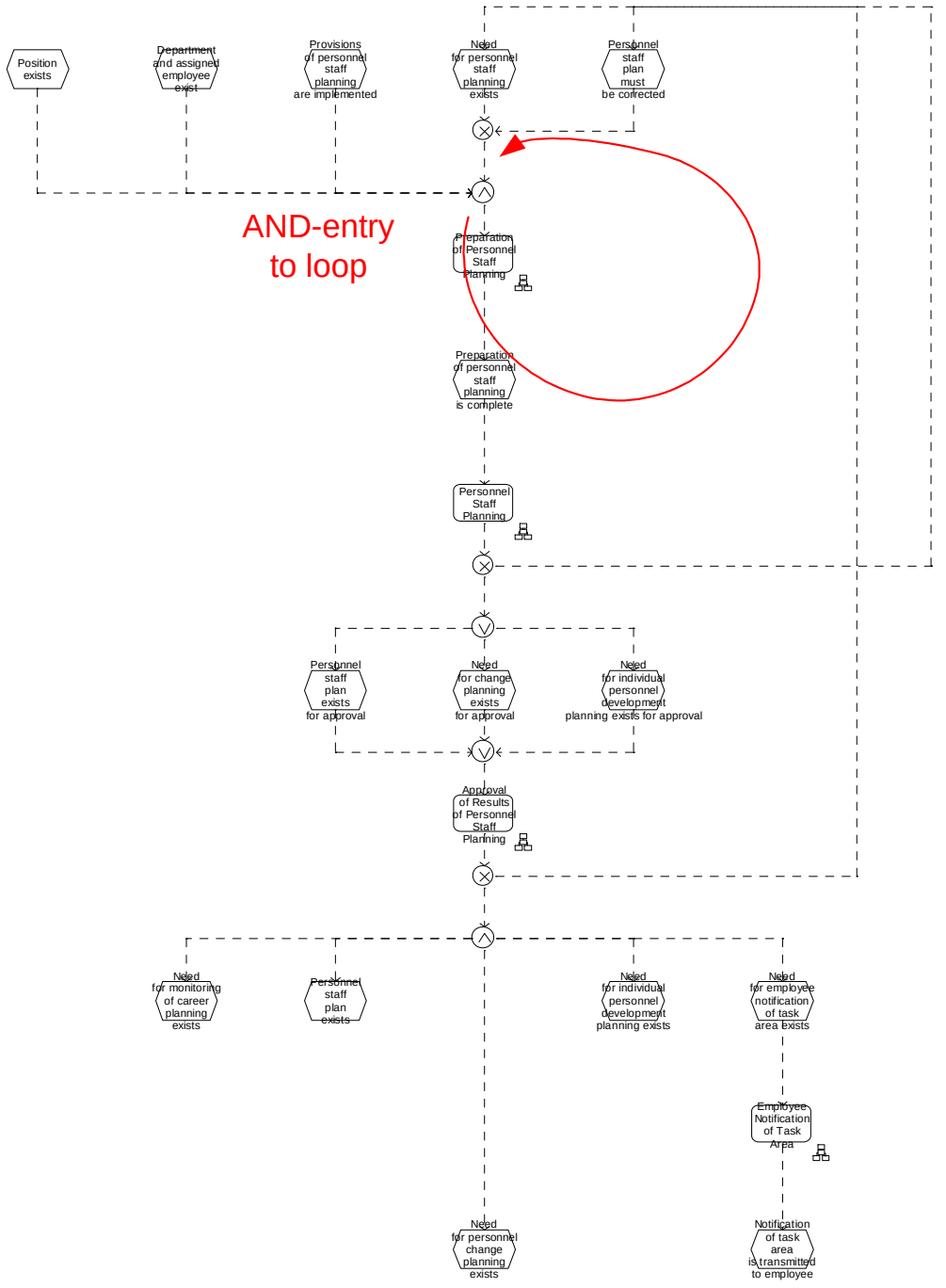


Figure A.34: Organizational Management – Planning Staff Assignment and Changes – Personnel Staff Planning (completely reduced)

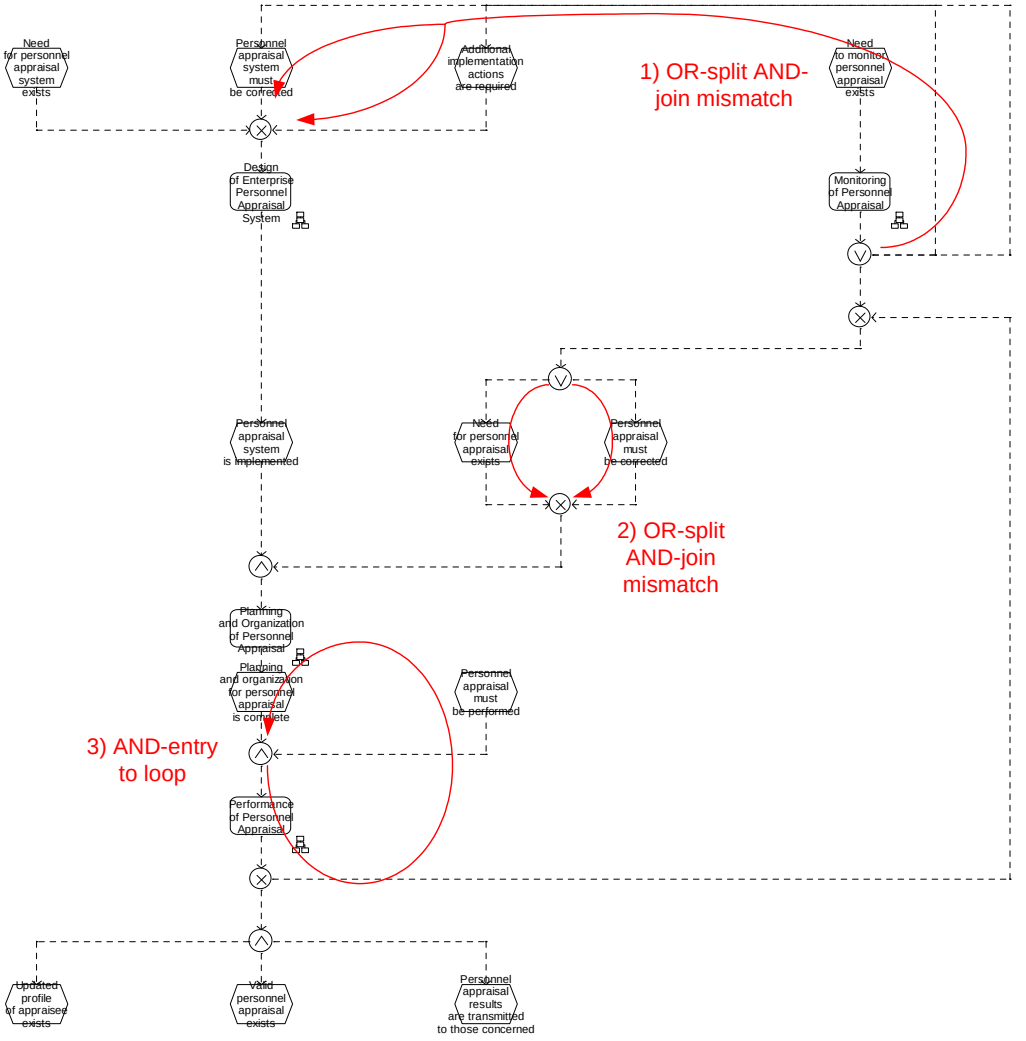


Figure A.35: Personnel Development – Personnel Appraisal (reduced size 8)

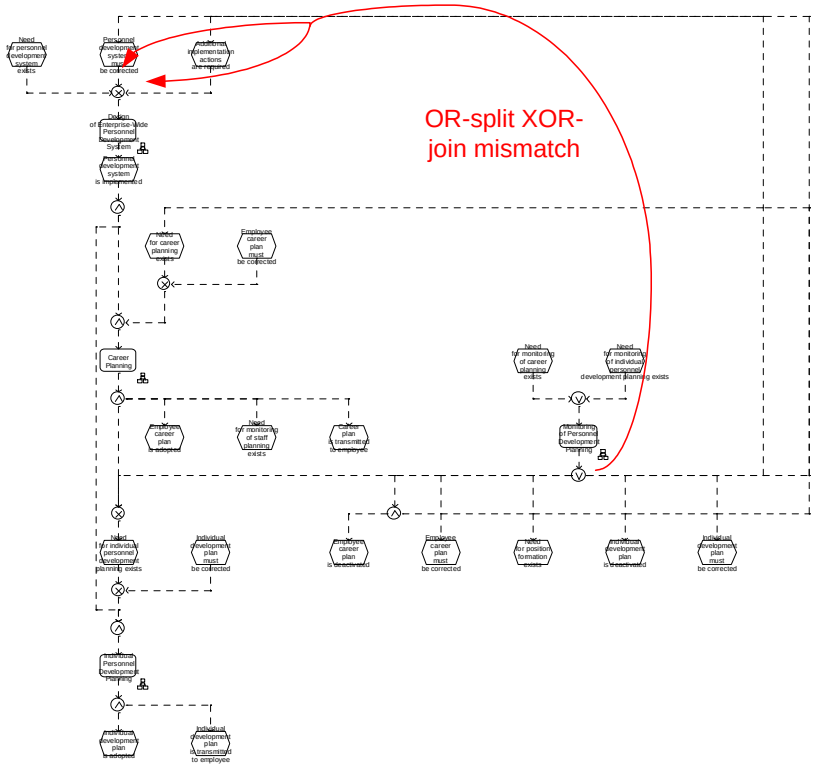


Figure A.36: Personnel Development – Personnel Development Planning (reduced size 13)

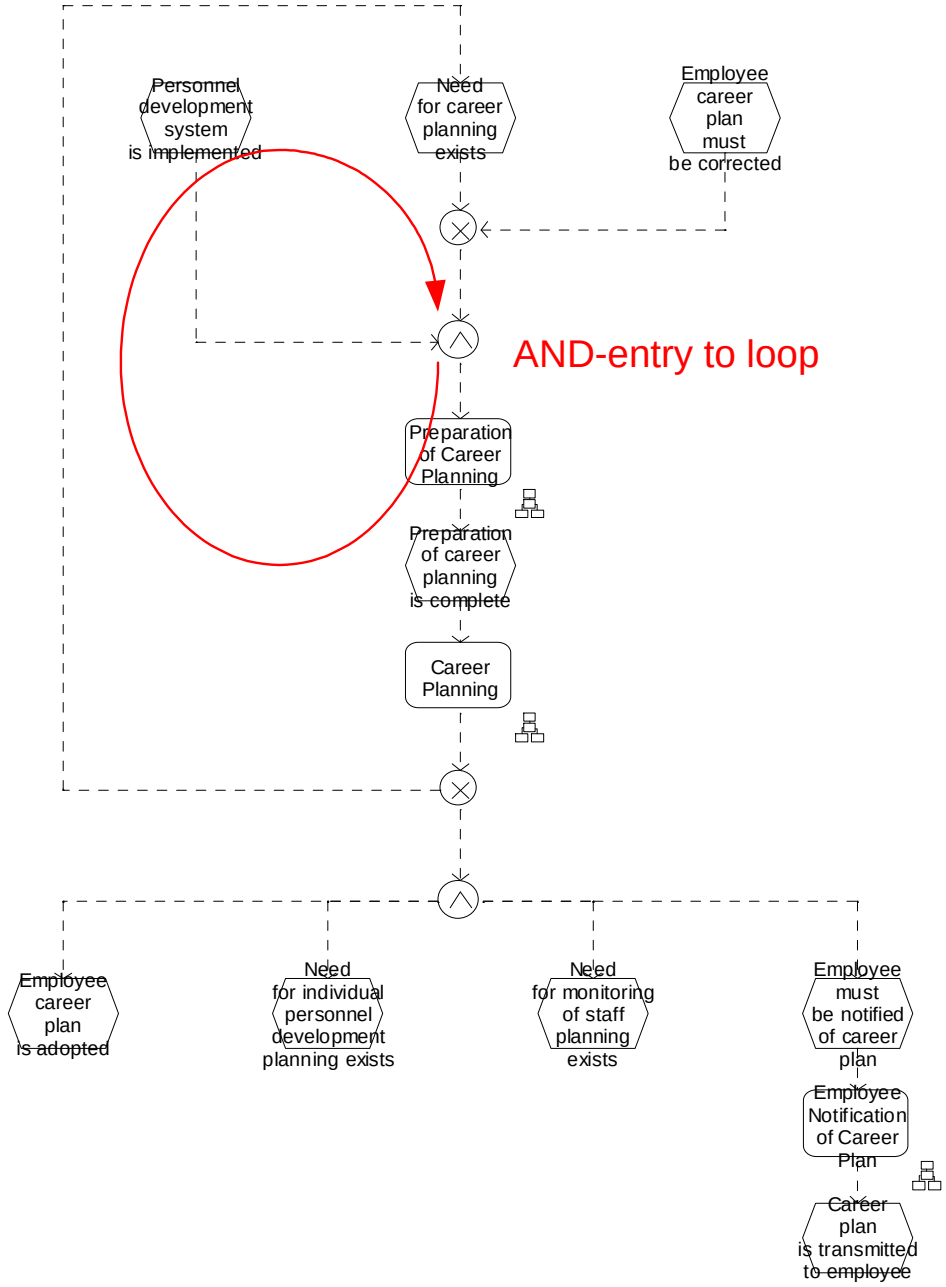


Figure A.37: Personnel Development – Personnel Development Planning – Career Planning (completely reduced)

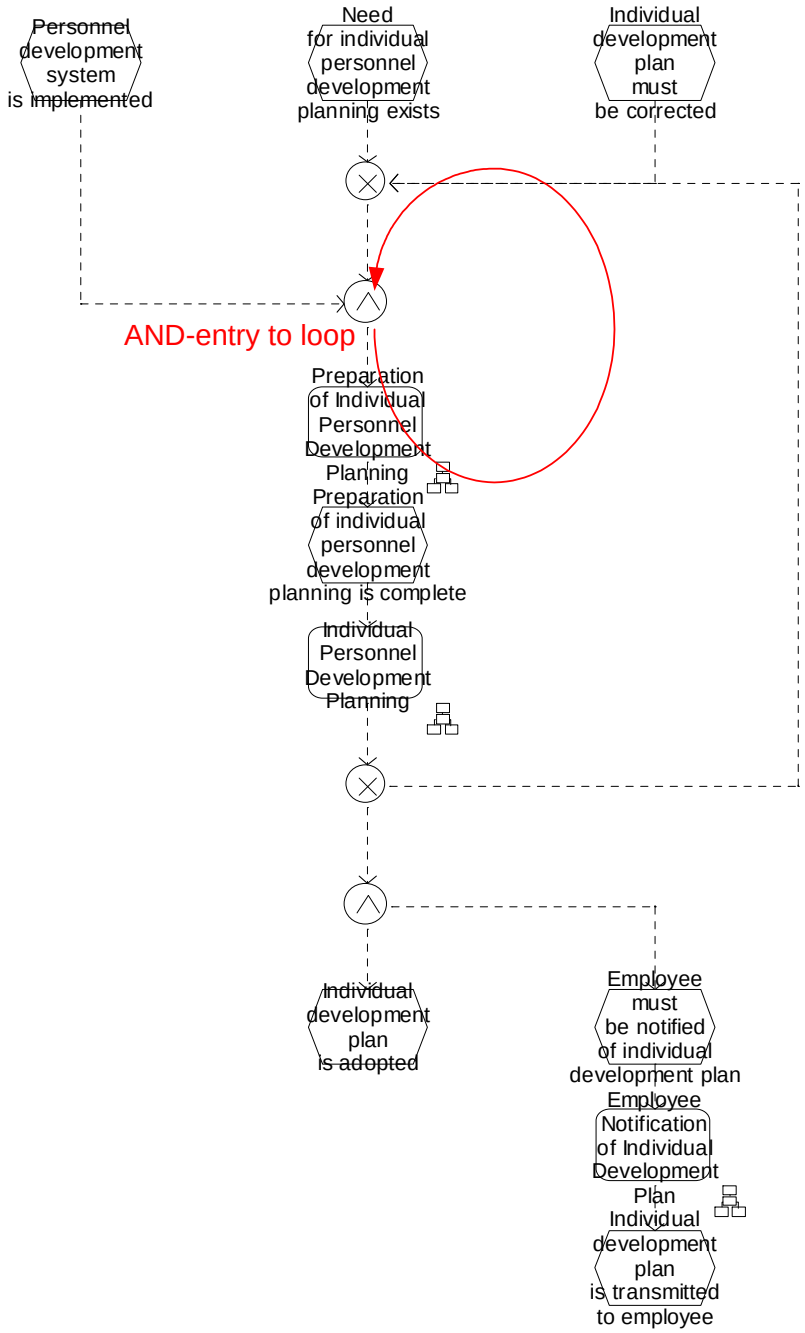


Figure A.38: Personnel Development – Personnel Development Planning – Individual Personnel Development Planning (completely reduced)

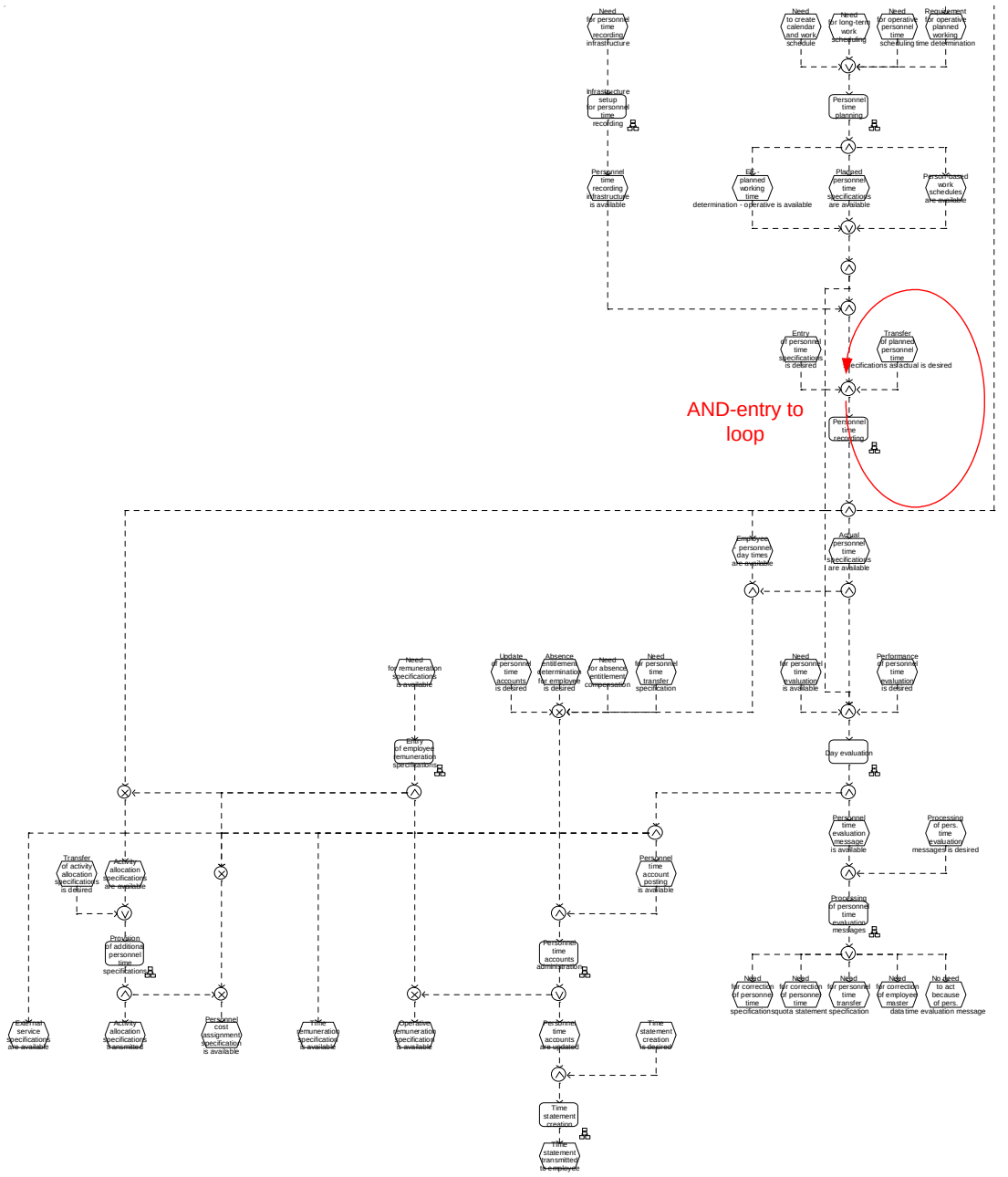


Figure A.39: Personnel Time Management – Personnel Time Management (reduced size 28)

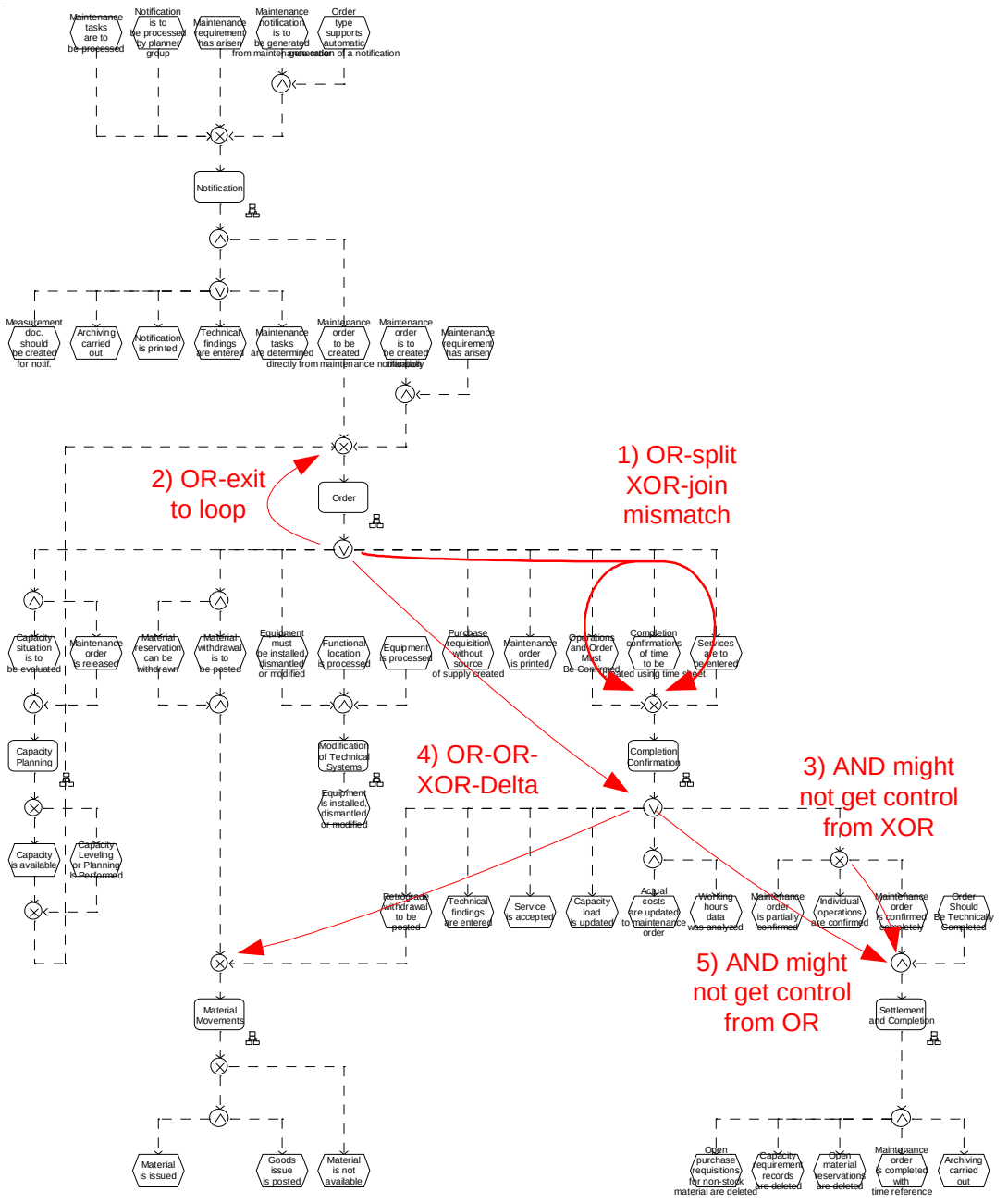


Figure A.40: Plant Maintenance – Breakdown Maintenance Processing (reduced size 9)

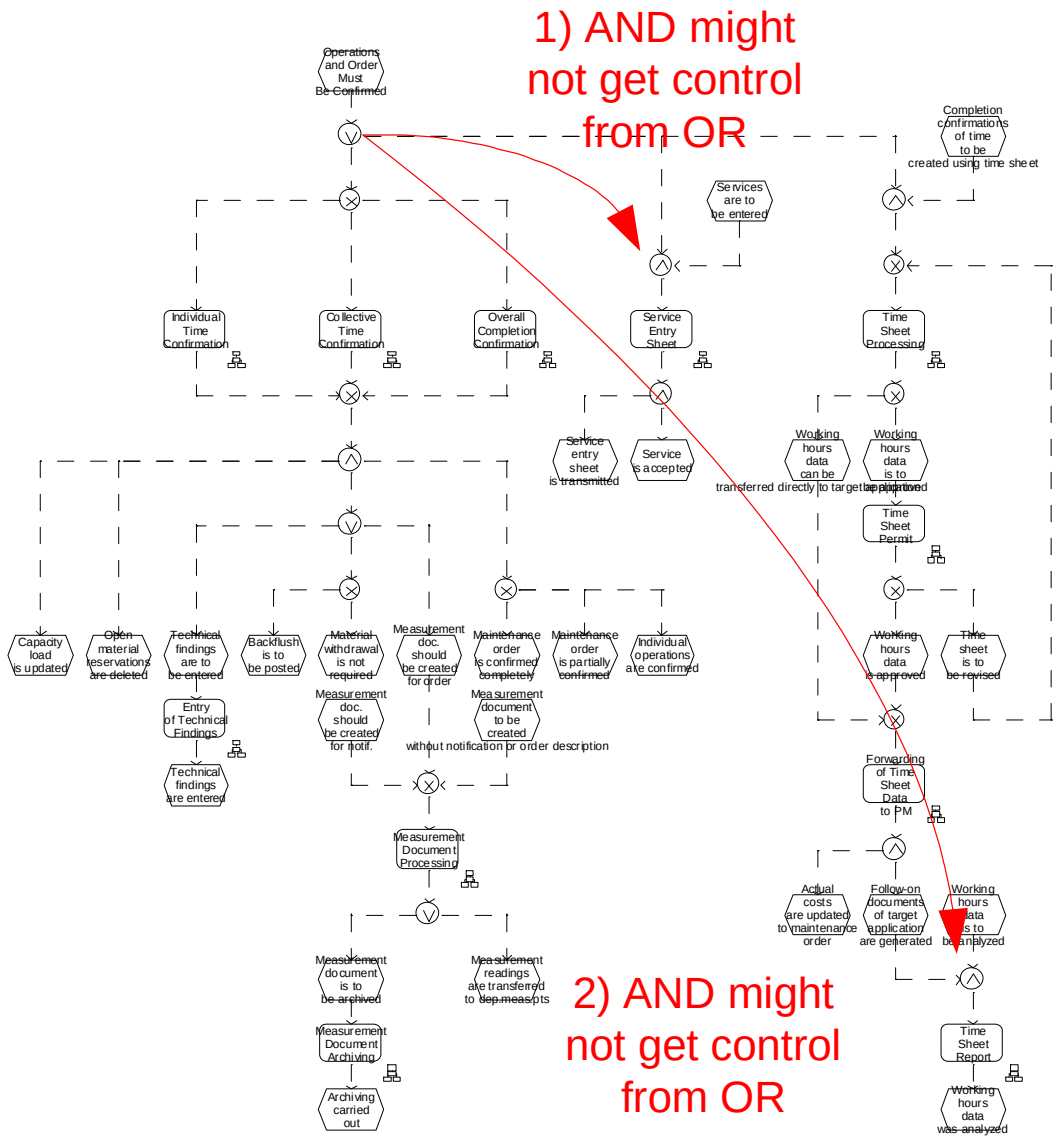


Figure A.41: Plant Maintenance – Breakdown Maintenance Processing – Completion Confirmation (reduced size 11)

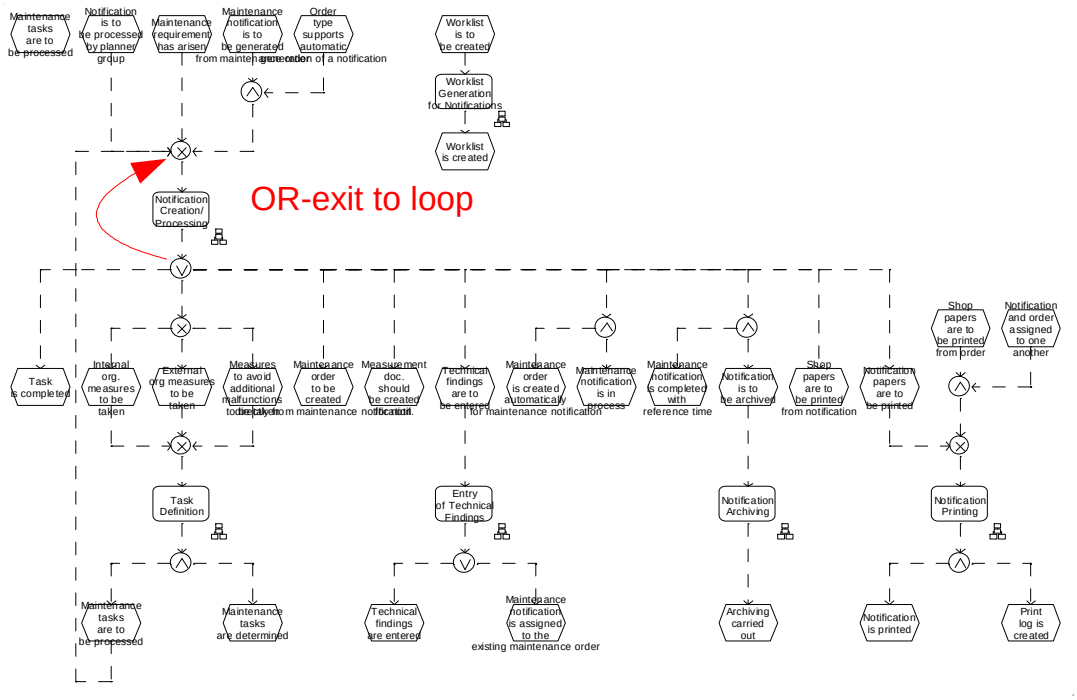


Figure A.42: Plant Maintenance – Breakdown Maintenance Processing – Notification (reduced size 6)

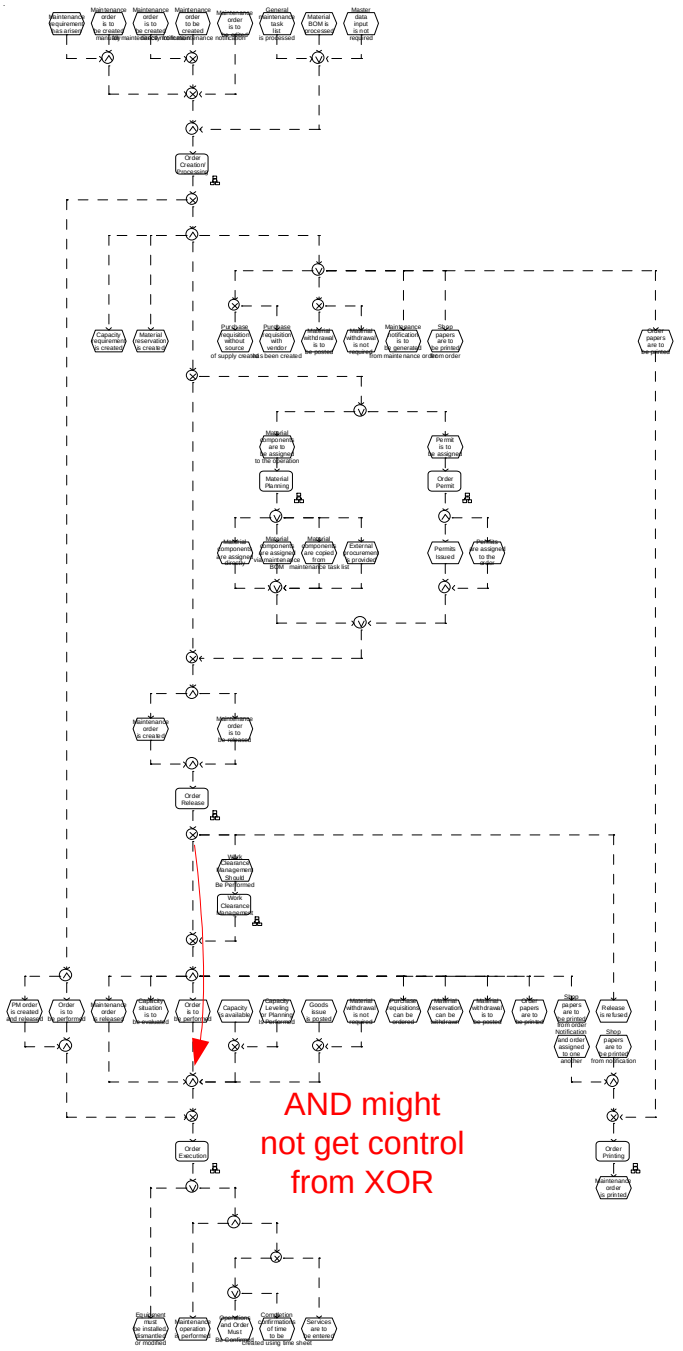


Figure A.43: Plant Maintenance – Breakdown Maintenance Processing – Order (reduced size 12)

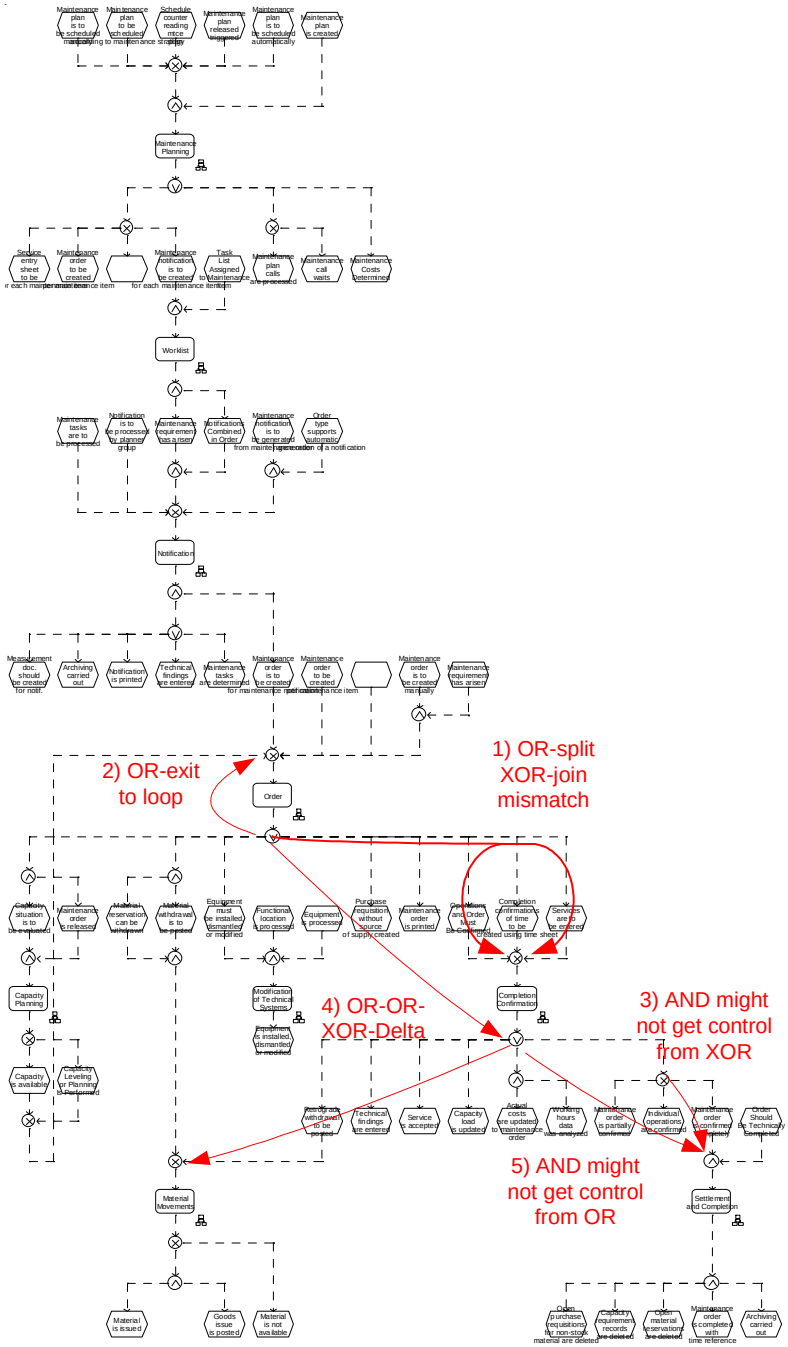


Figure A.44: Plant Maintenance – Planned Maintenance Processing (reduced size 9)

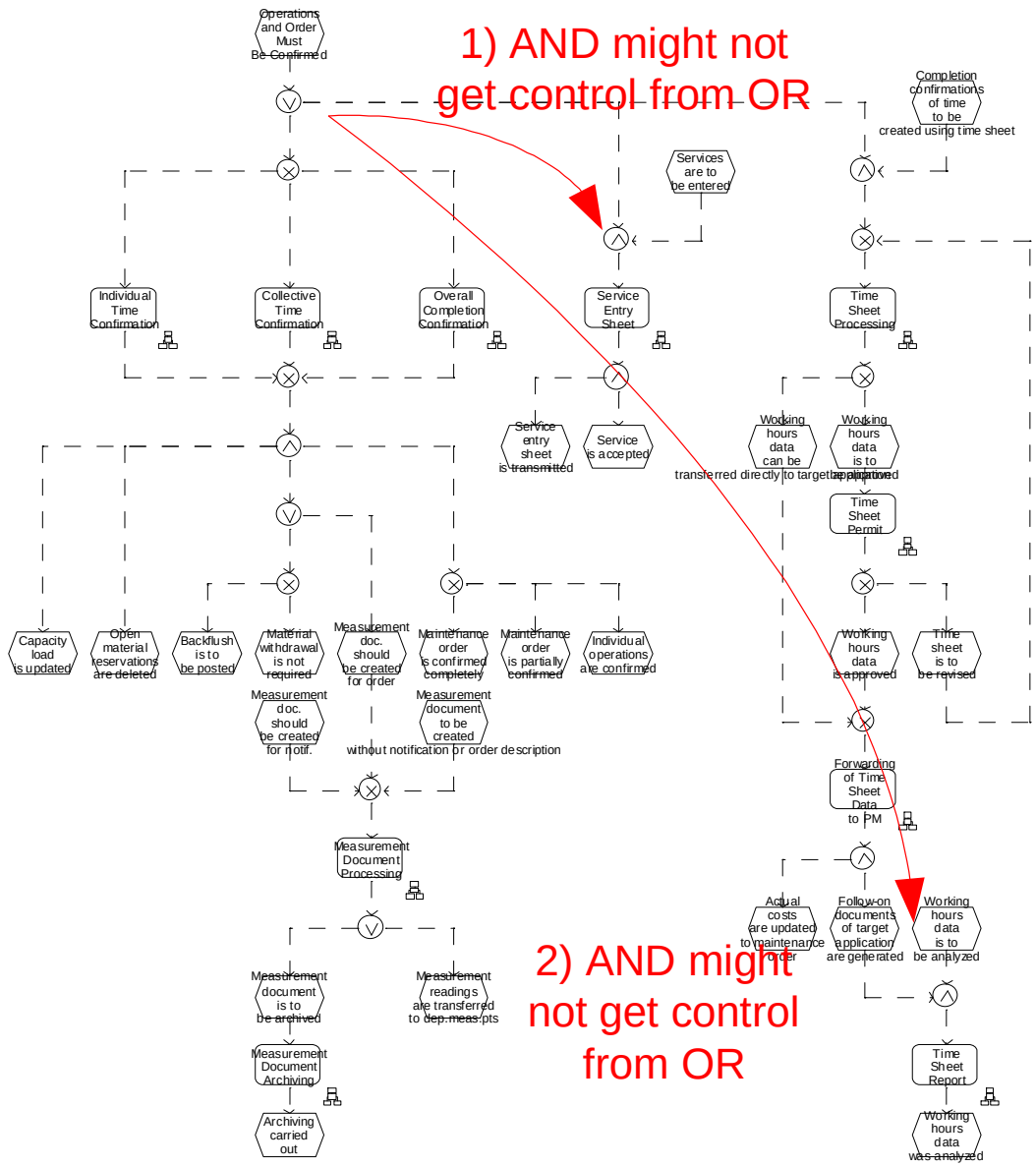


Figure A.45: Plant Maintenance – Planned Maintenance Processing – Completion Confirmation (reduced size 11)

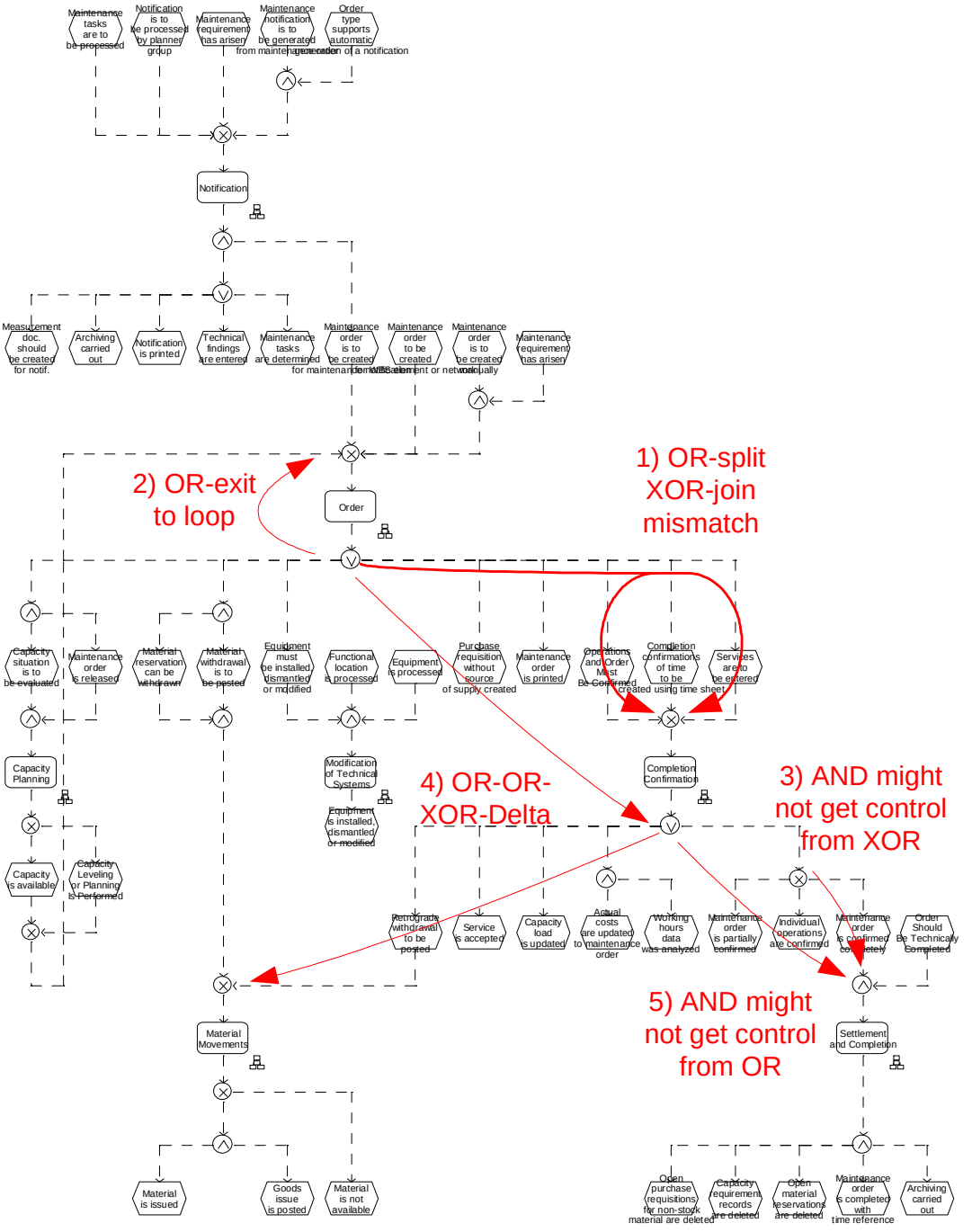


Figure A.46: Plant Maintenance – Project-Based Maintenance Processing (reduced size 9)

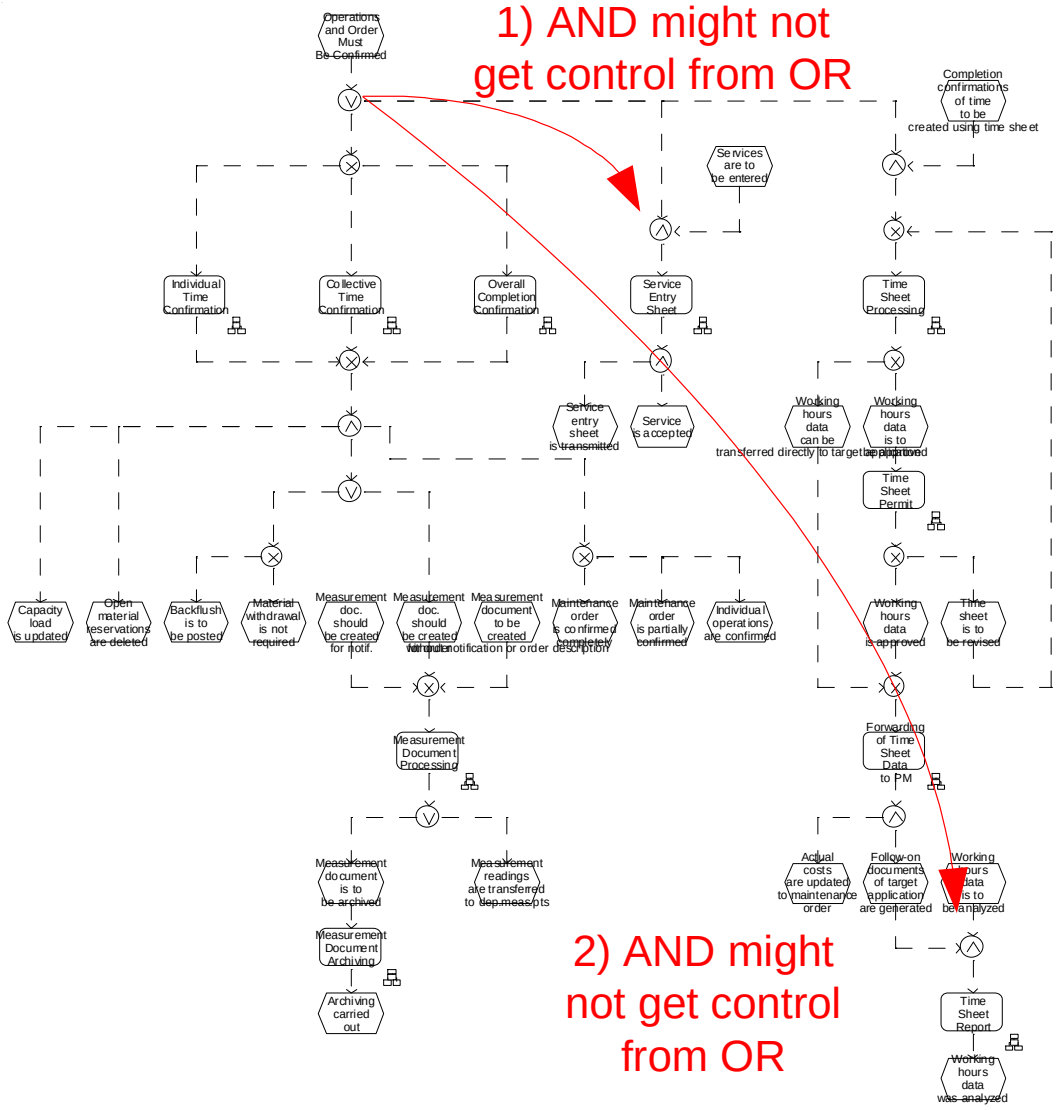


Figure A.47: Plant Maintenance – Project-Based Maintenance Processing – Completion Confirmation (reduced size 11)

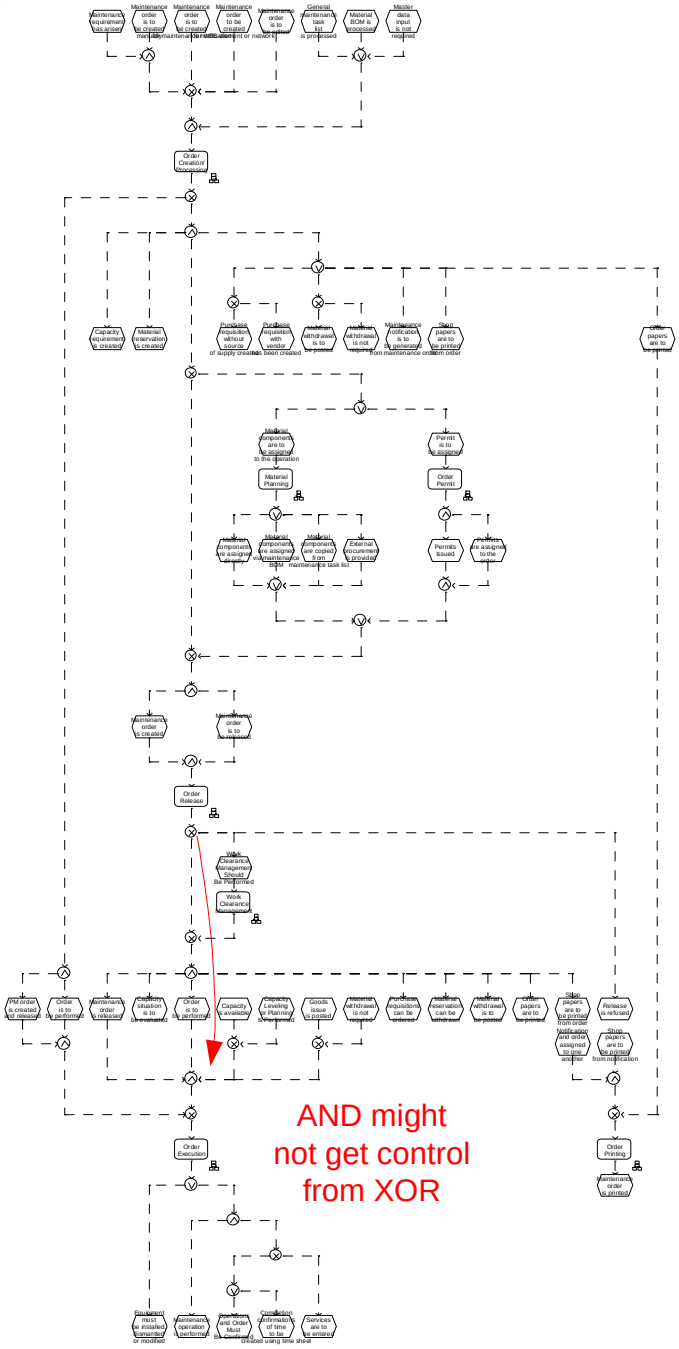


Figure A.48: Plant Maintenance – Project-Based Maintenance Processing – Order (reduced size 12)

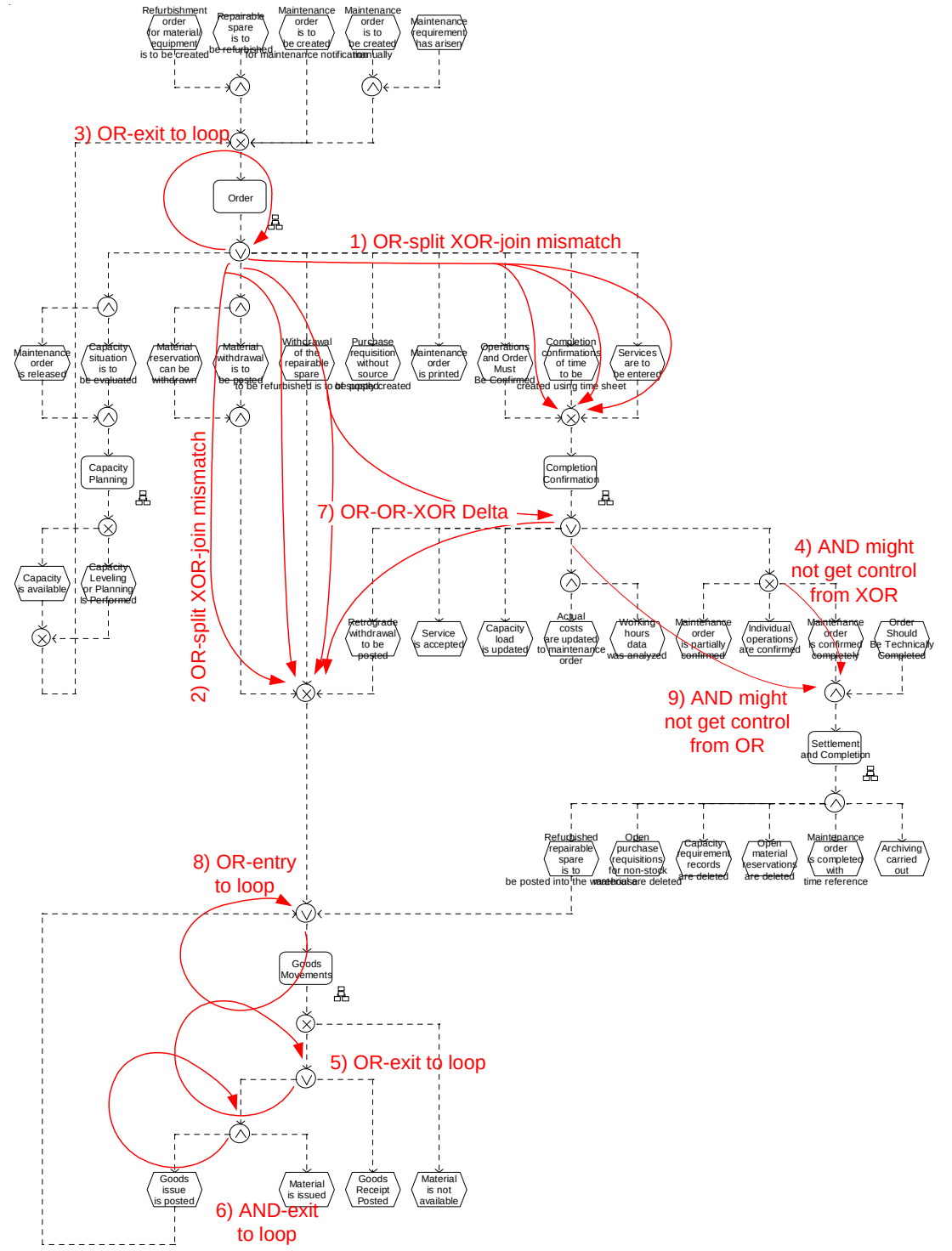


Figure A.49: Plant Maintenance – Refurbishment Processing in Plant Maintenance (reduced size 7)

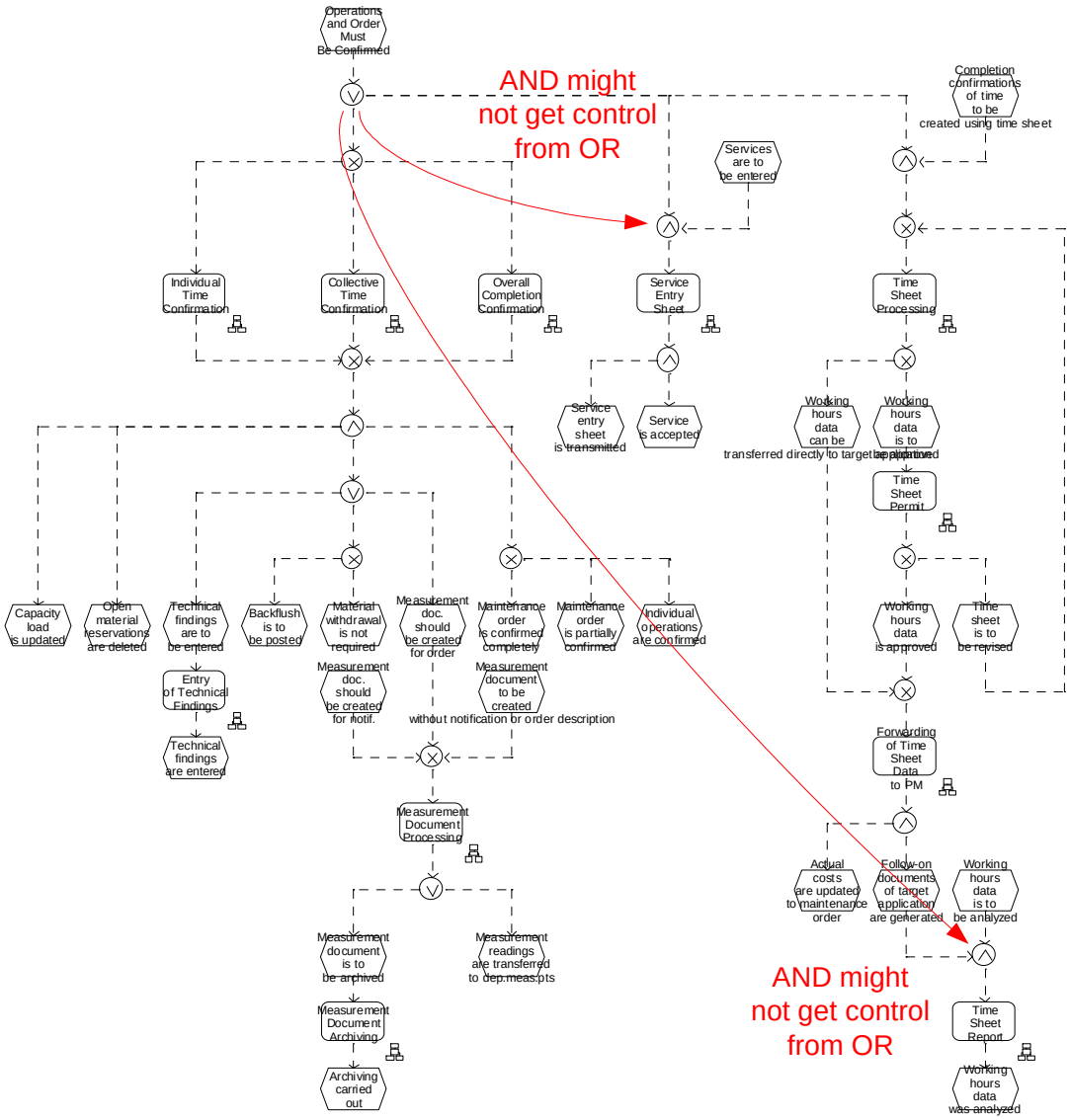


Figure A.50: Plant Maintenance – Refurbishment Processing in Plant Maintenance – Completion Confirmation (reduced size 11)

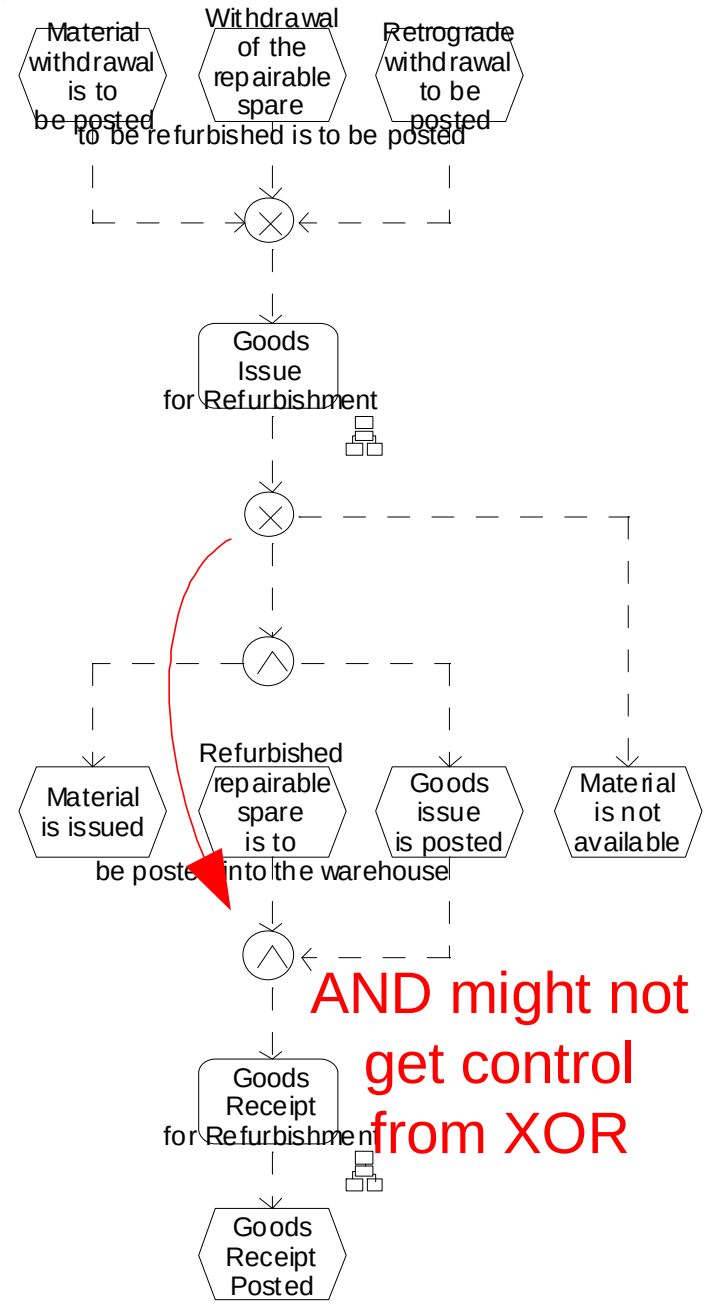


Figure A.51: Plant Maintenance – Refurbishment Processing in Plant Maintenance – Goods Movements (completely reduced)

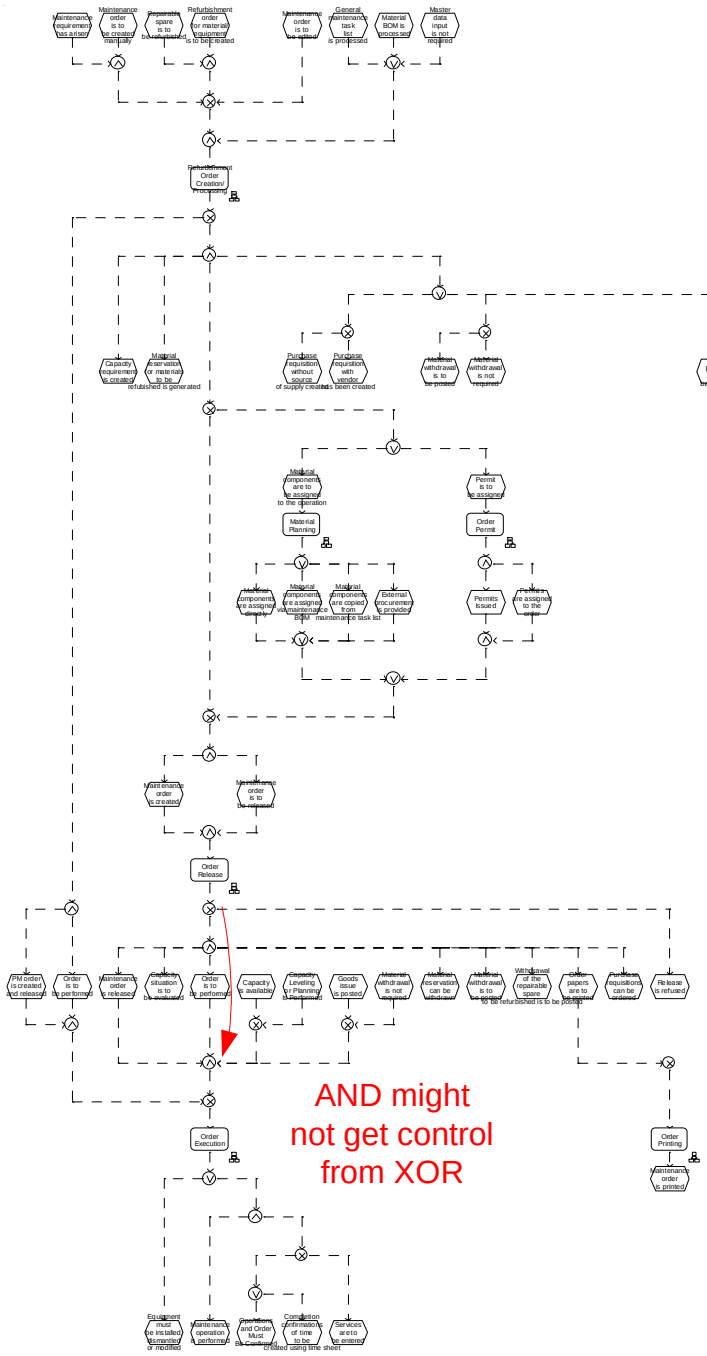


Figure A.52: Plant Maintenance – Refurbishment Processing in Plant Maintenance – Order (reduced size 6)

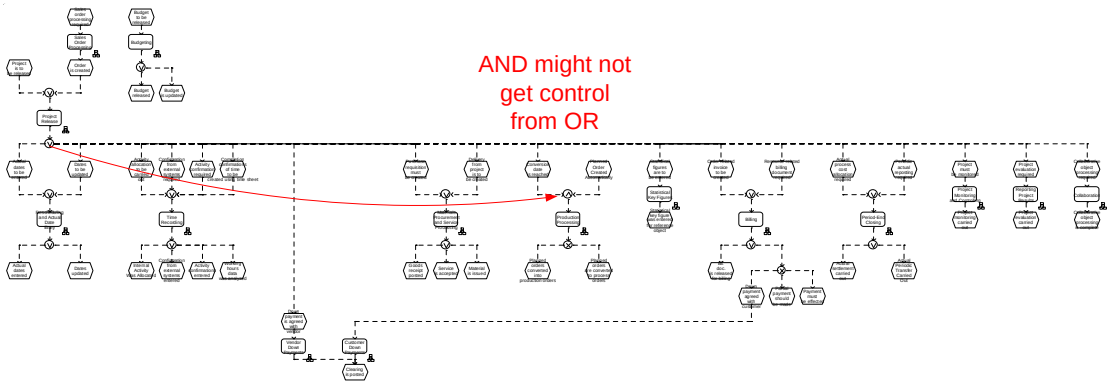


Figure A.53: Project Management – Execution (completely reduced)

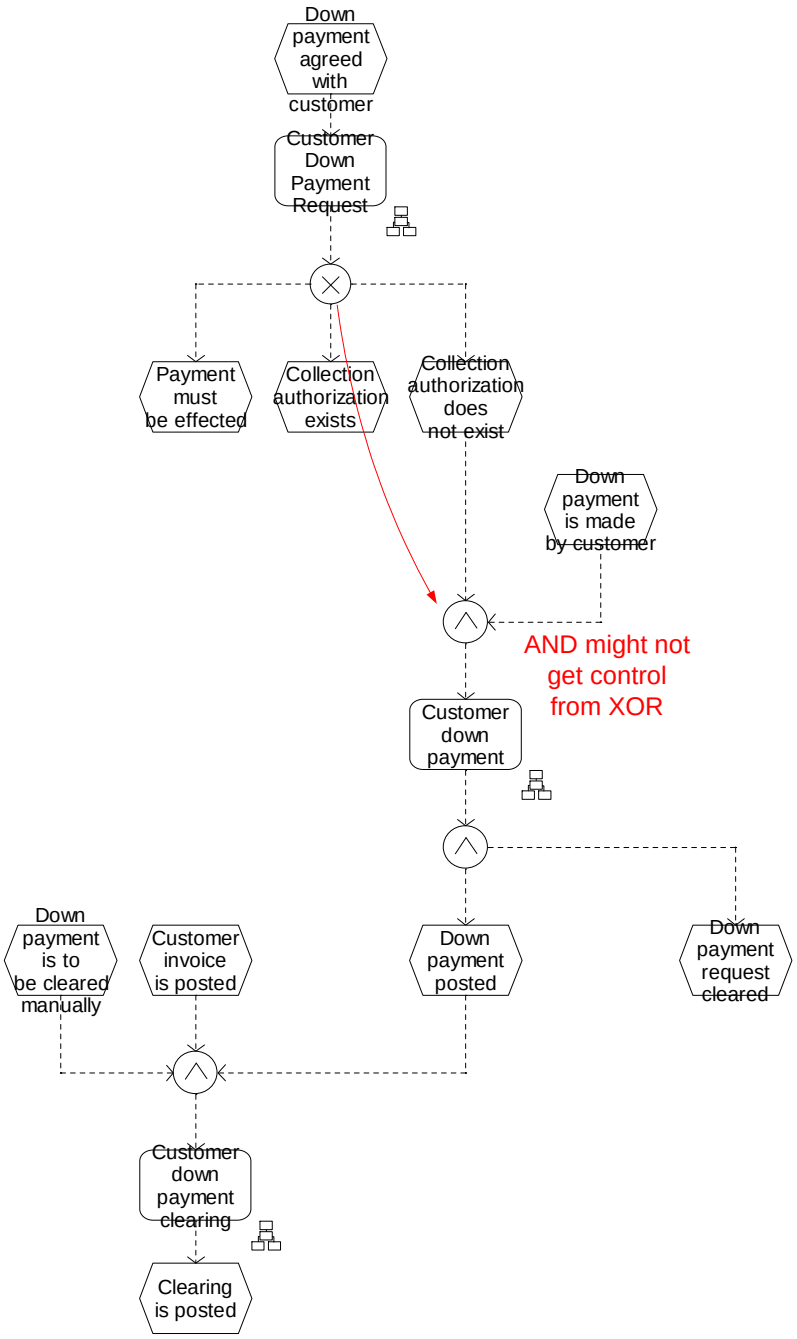


Figure A.54: Project Management – Execution – Customer Down Payments (completely reduced)

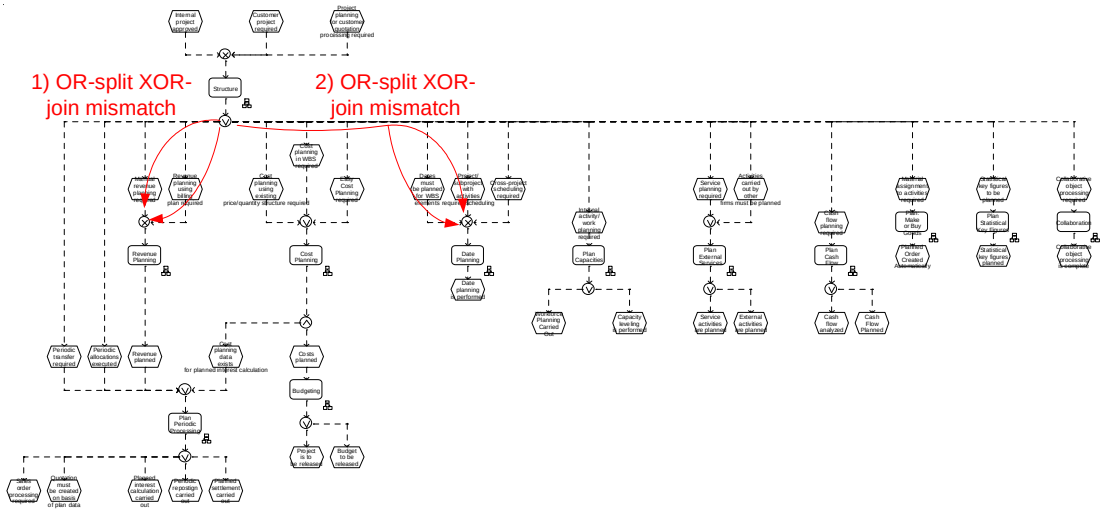


Figure A.55: Project Management – Planning (completely reduced)

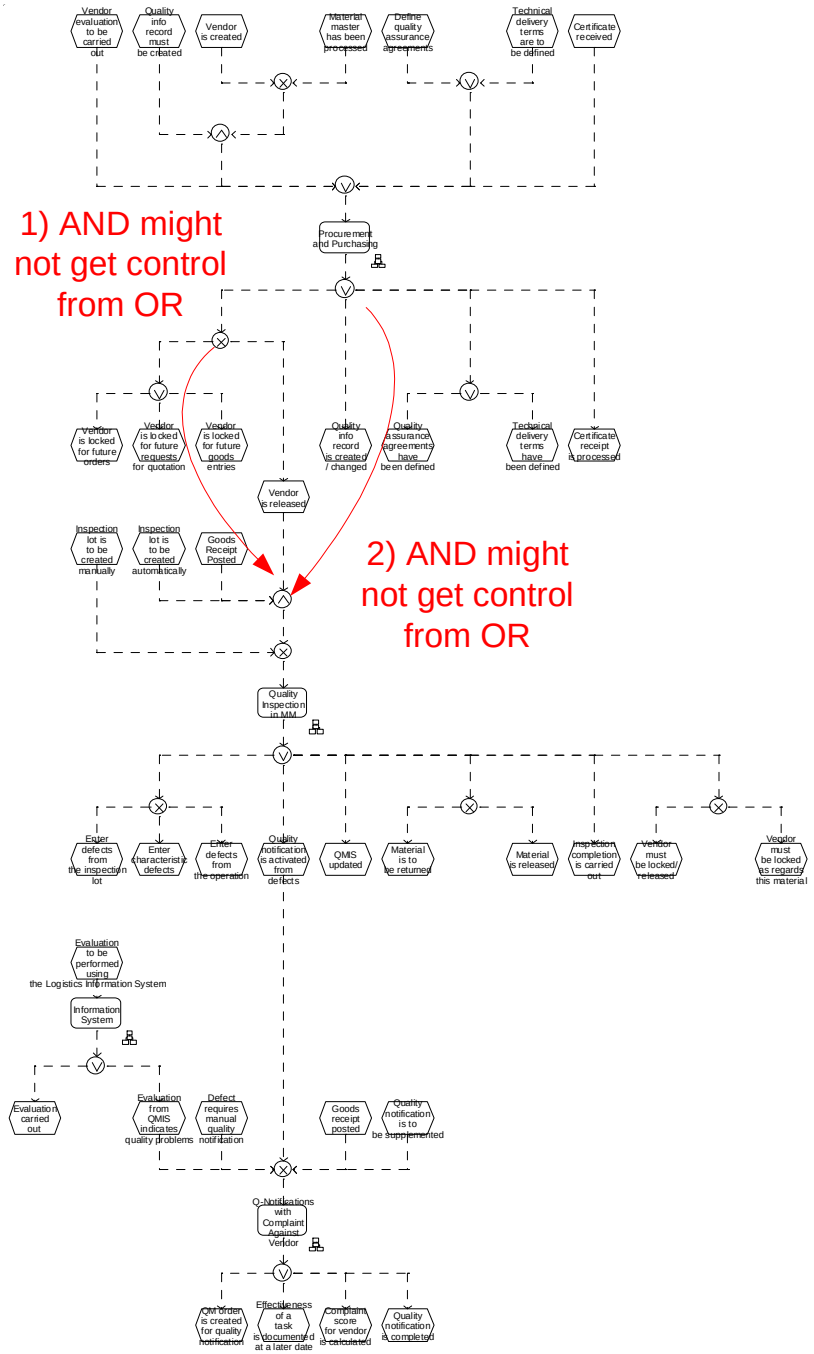


Figure A.56: Quality Management – QM in Materials Management (reduced size 9)

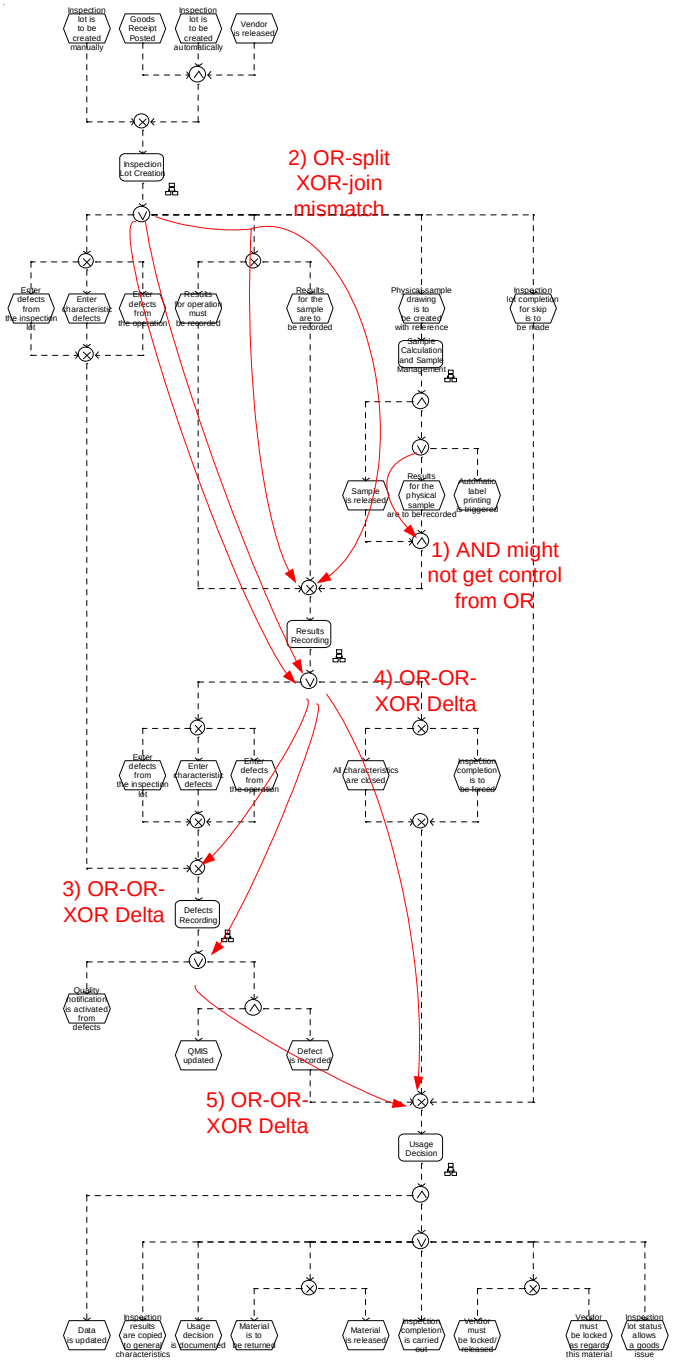


Figure A.57: Quality Management – QM in Materials Management – Quality Inspection in MM (completely reduced)

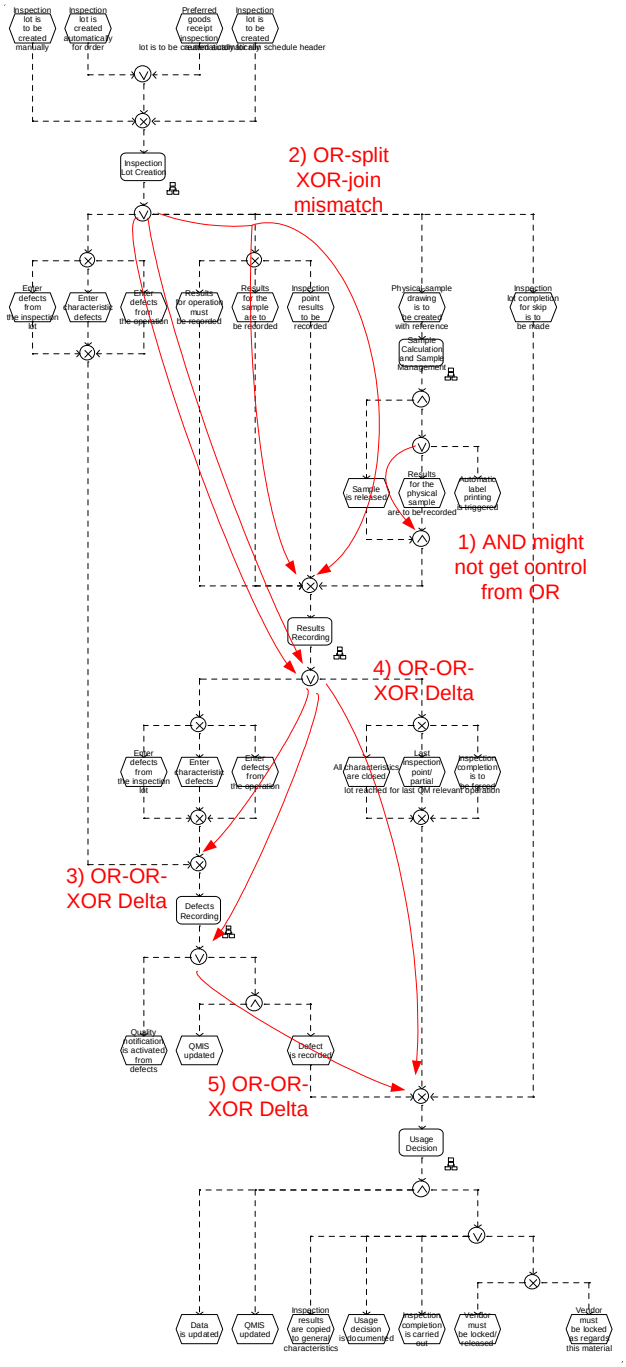


Figure A.58: Quality Management – QM in Production – Inspection During Production (completely reduced)

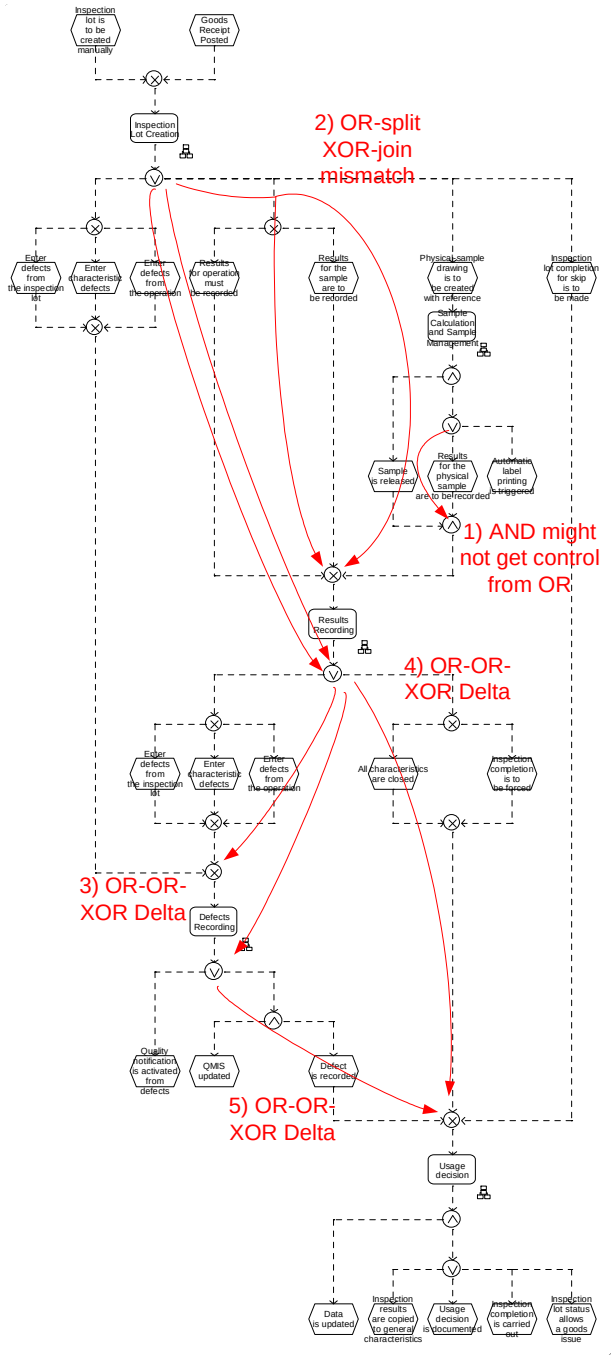


Figure A.59: Quality Management – QM in Production – Quality Inspection for Goods Receipt from Production (completely reduced)

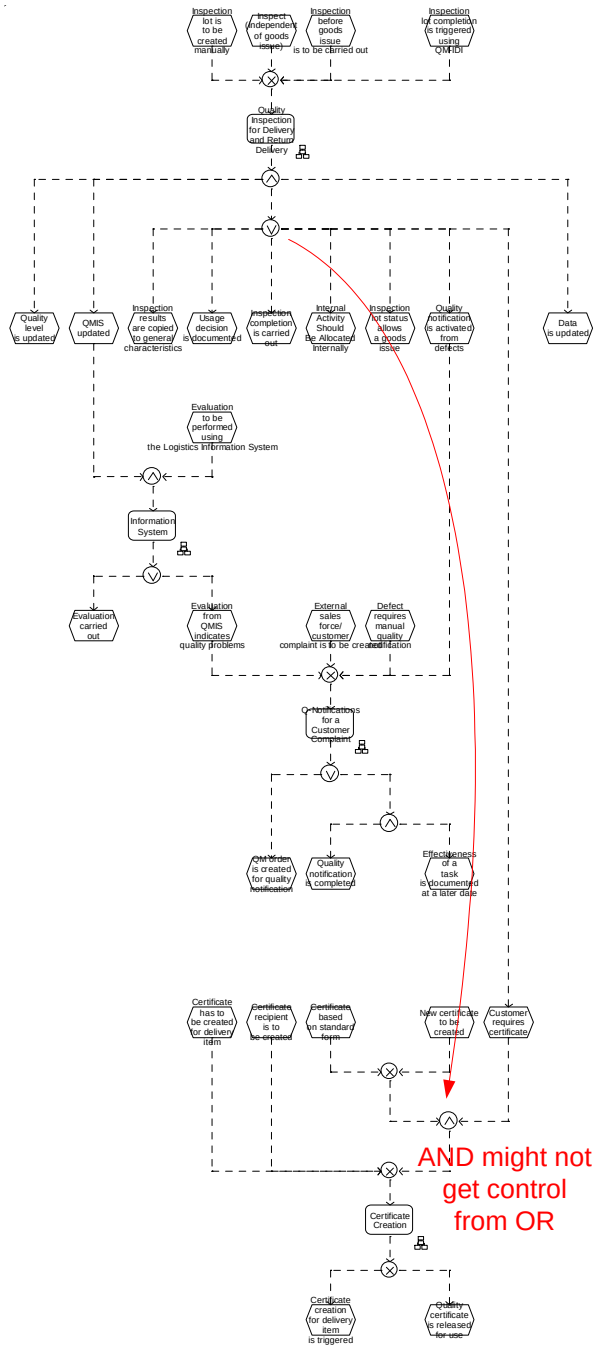


Figure A.60: Quality Management – QM in Sales and Distribution (reduced size 15)

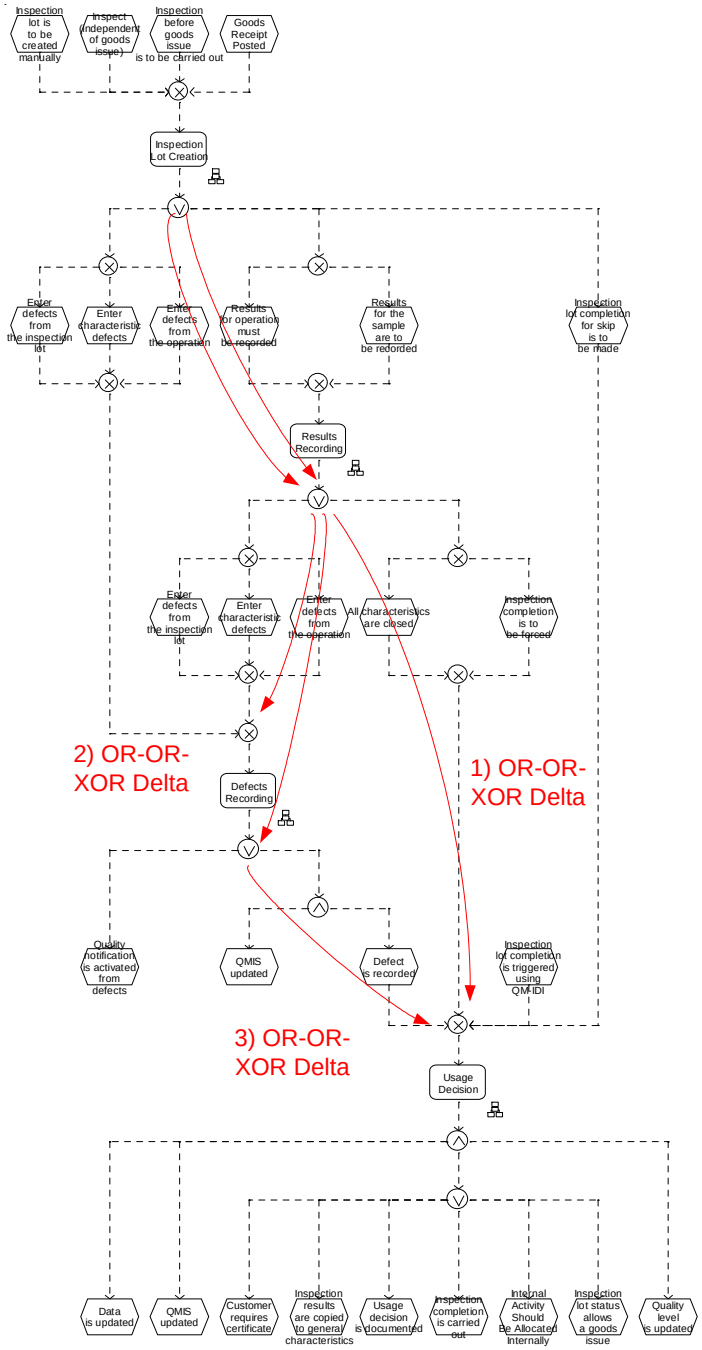


Figure A.61: Quality Management – QM in Sales and Distribution – Quality Inspection for Delivery and Return Delivery (reduced size 6)

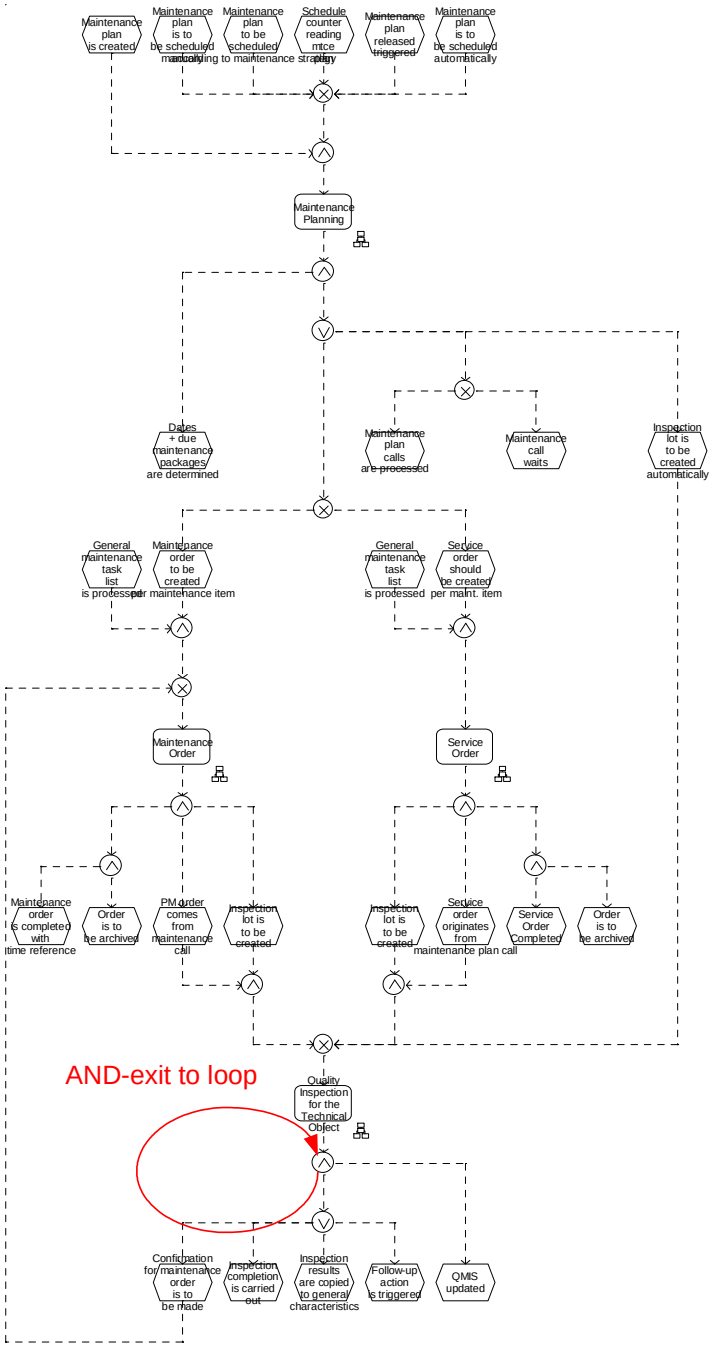


Figure A.62: Quality Management – Test Equipment Management (reduced size 14)

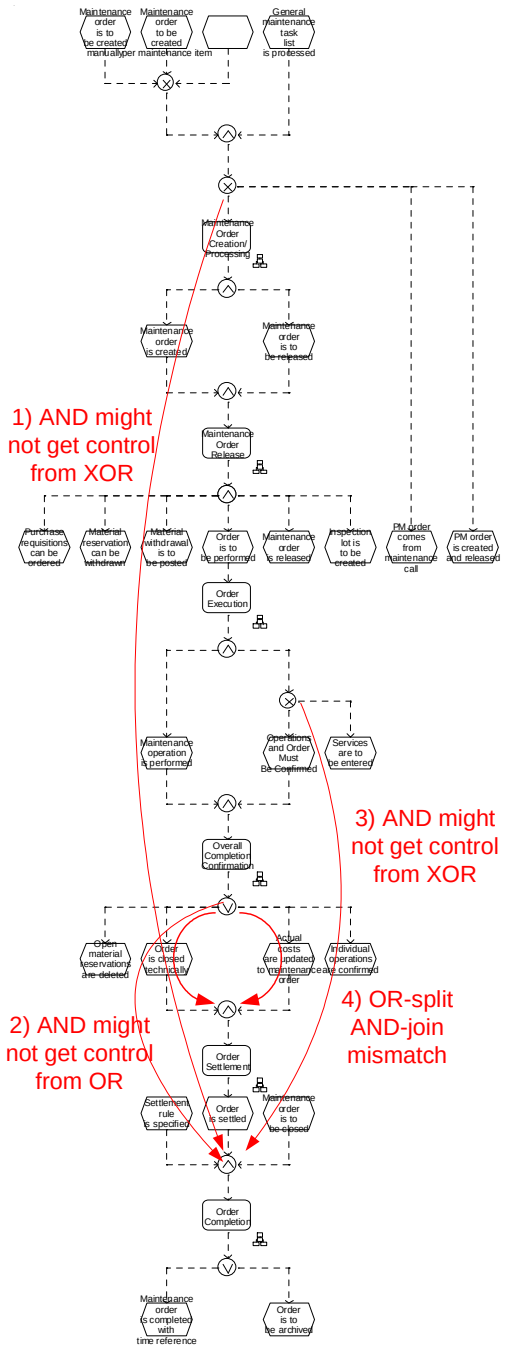


Figure A.63: Quality Management – Test Equipment Management – Maintenance Order (completely reduced)

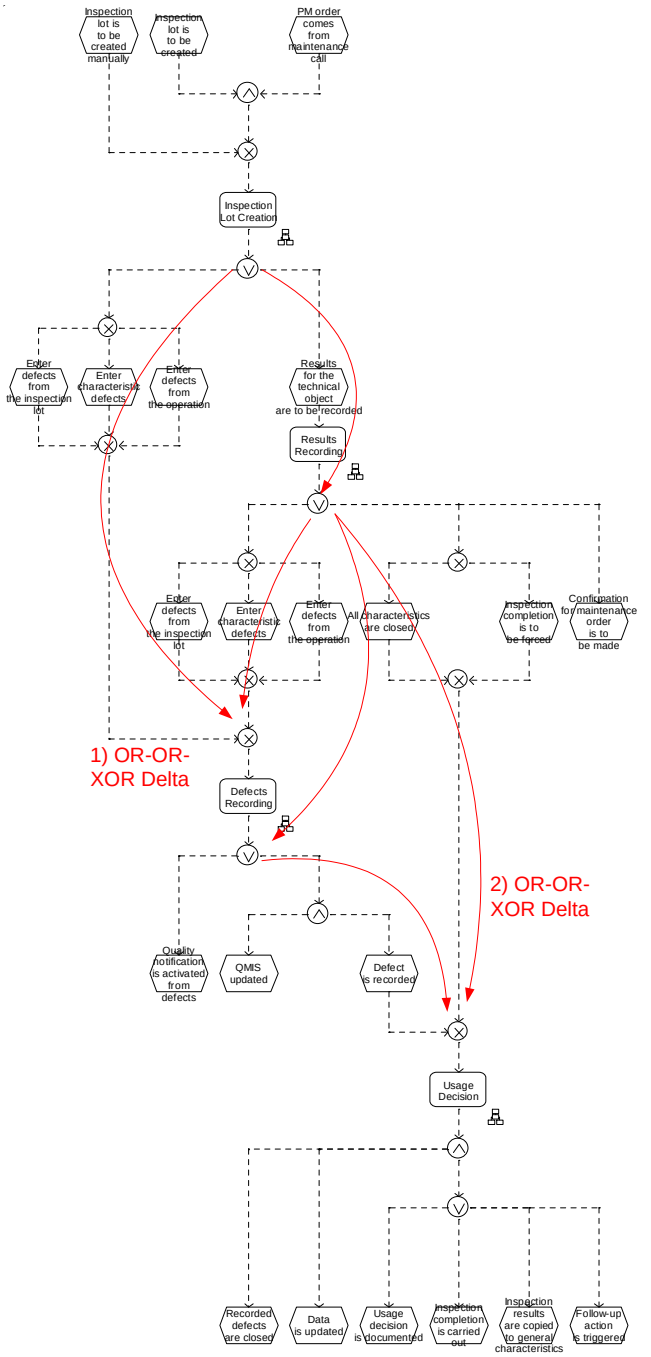


Figure A.64: Quality Management – Test Equipment Management – Quality Inspection for the Technical Object (completely reduced)

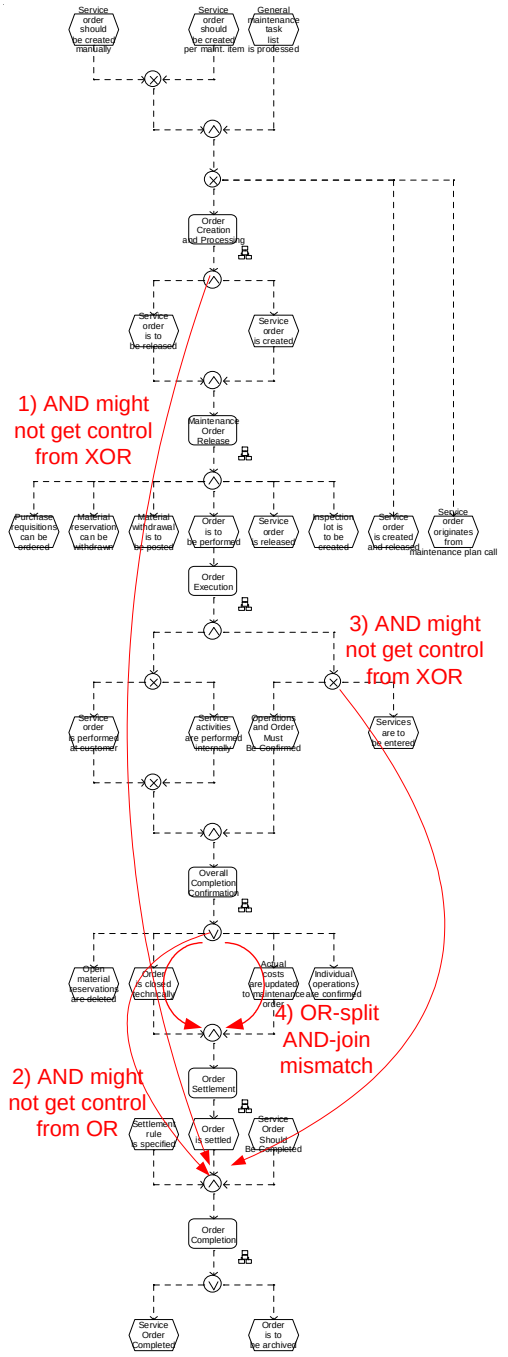


Figure A.65: Quality Management – Test Equipment Management – Service Order (completely reduced)

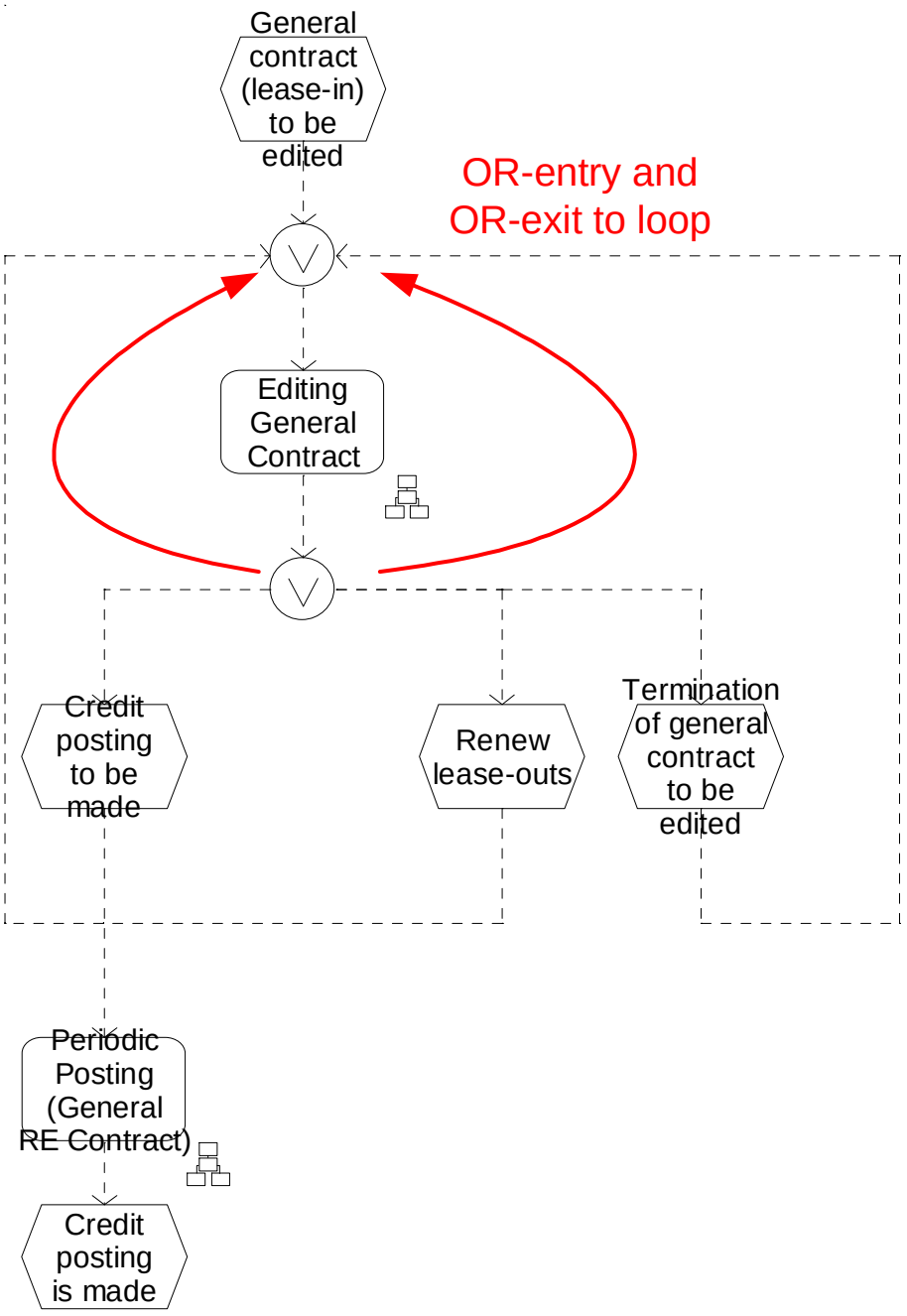


Figure A.66: Real Estate Management – Real Estate Management – General Contract (completely reduced)

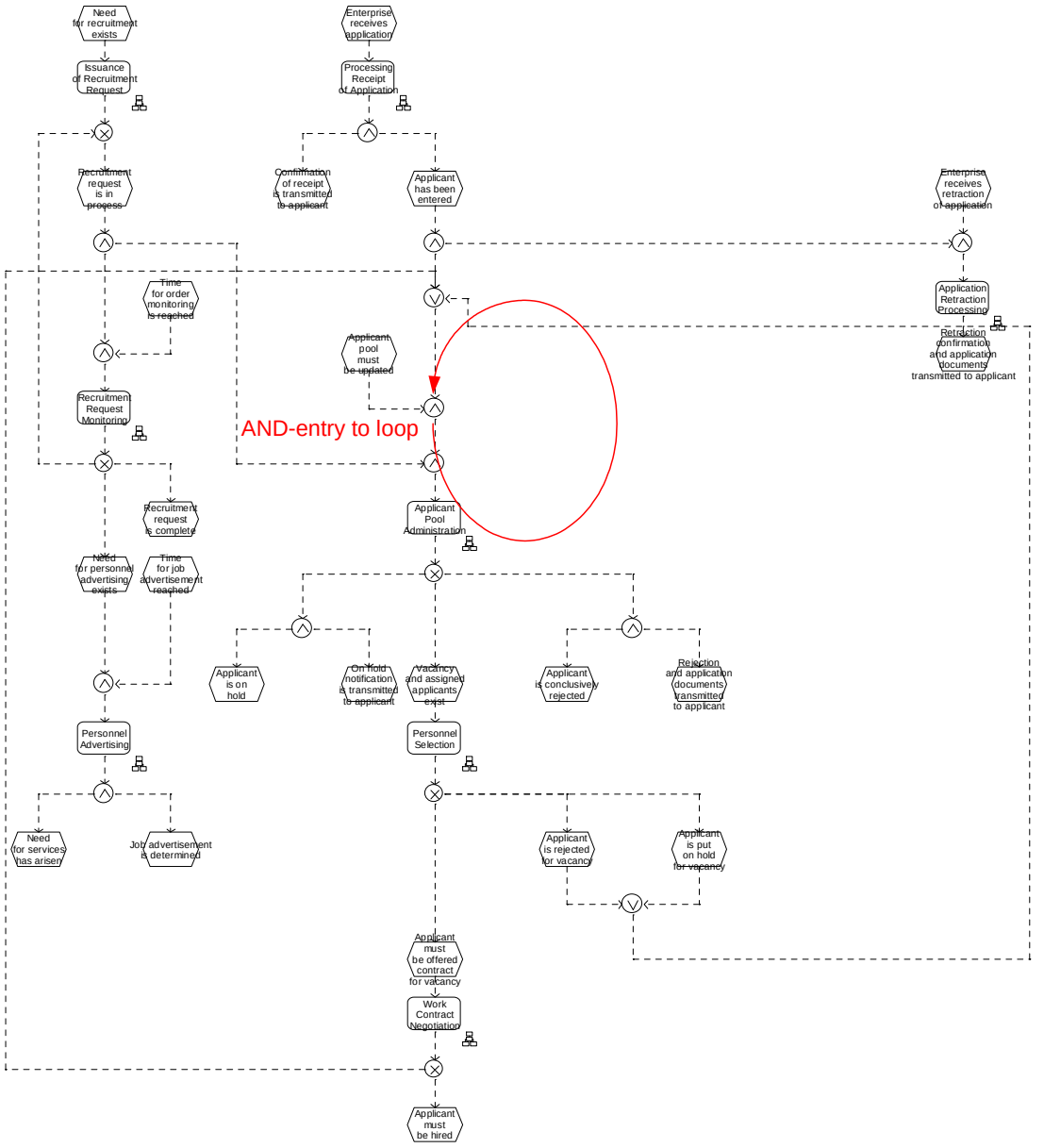


Figure A.67: Recruitment – Recruitment (reduced size 19)

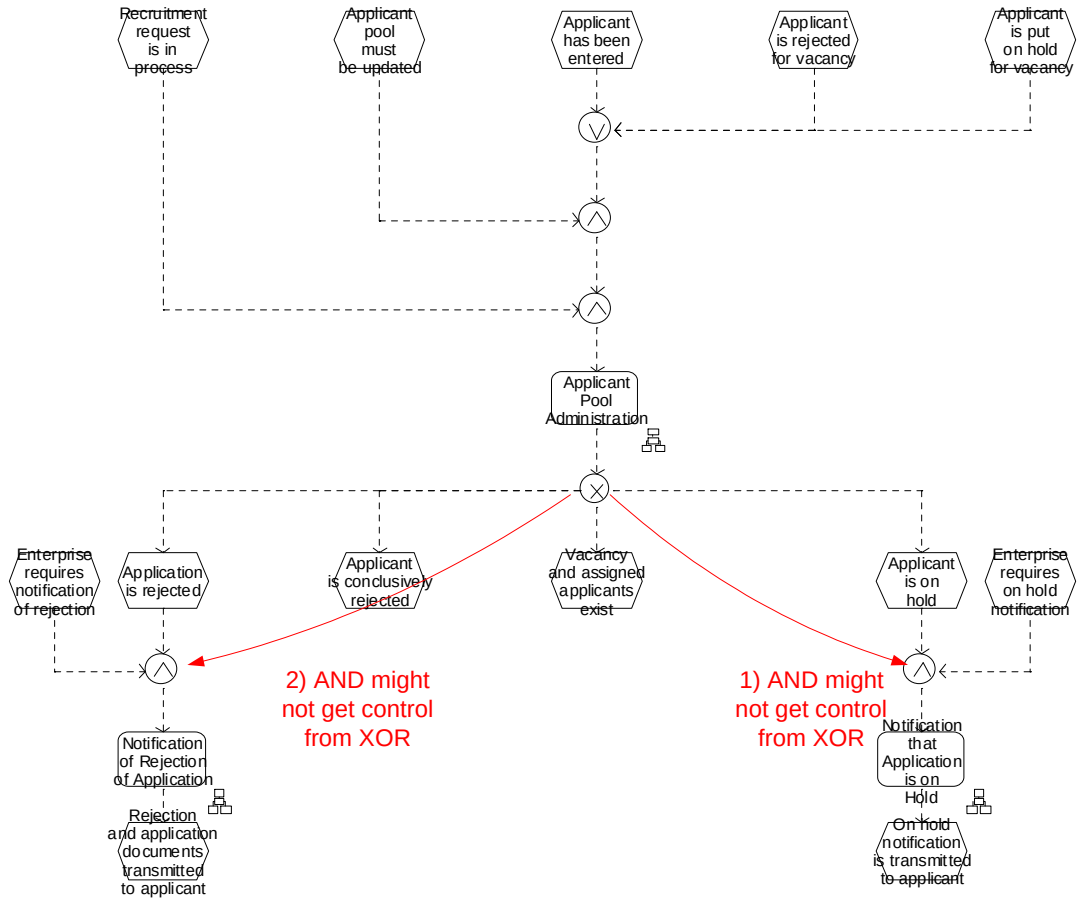


Figure A.68: Recruitment – Recruitment – Applicant Pool Administration (reduced size 8)

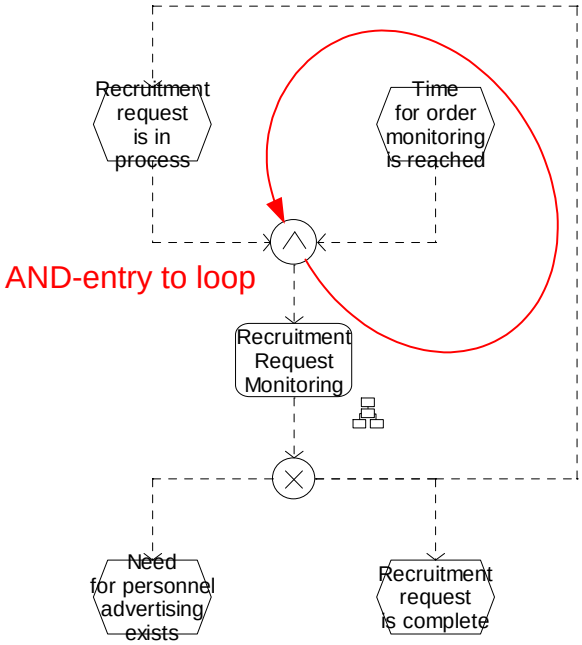


Figure A.69: Recruitment – Recruitment – Recruitment Request Monitoring (completely reduced)

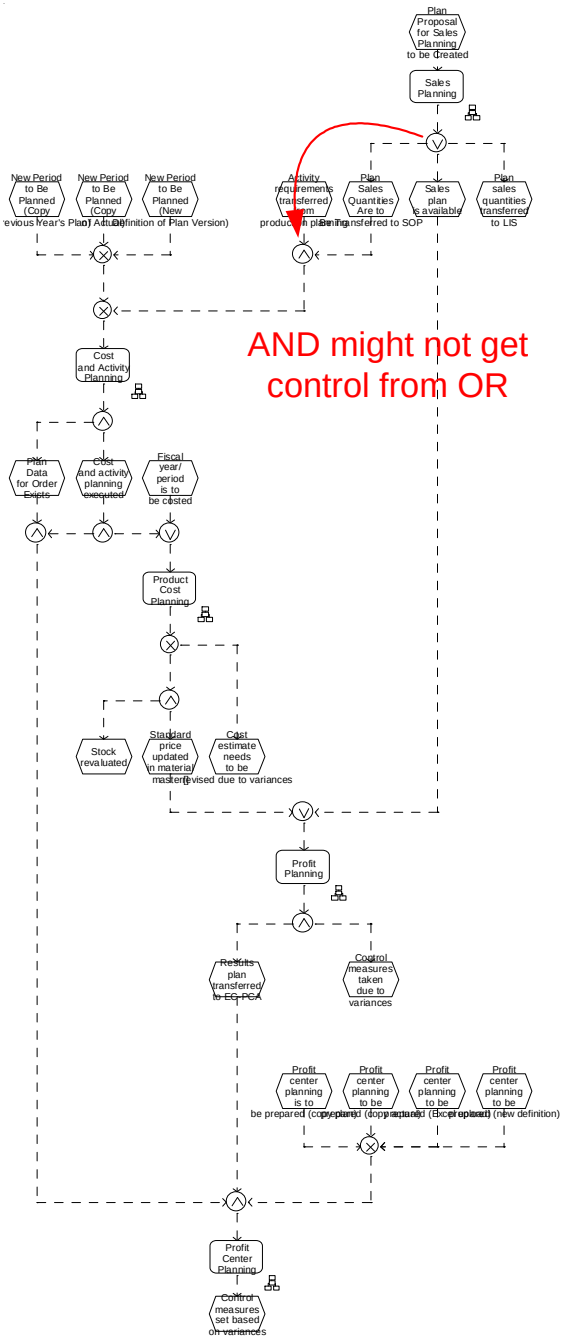


Figure A.70: Revenue and Cost Controlling – Profit and Cost Planning (reduced size 15)

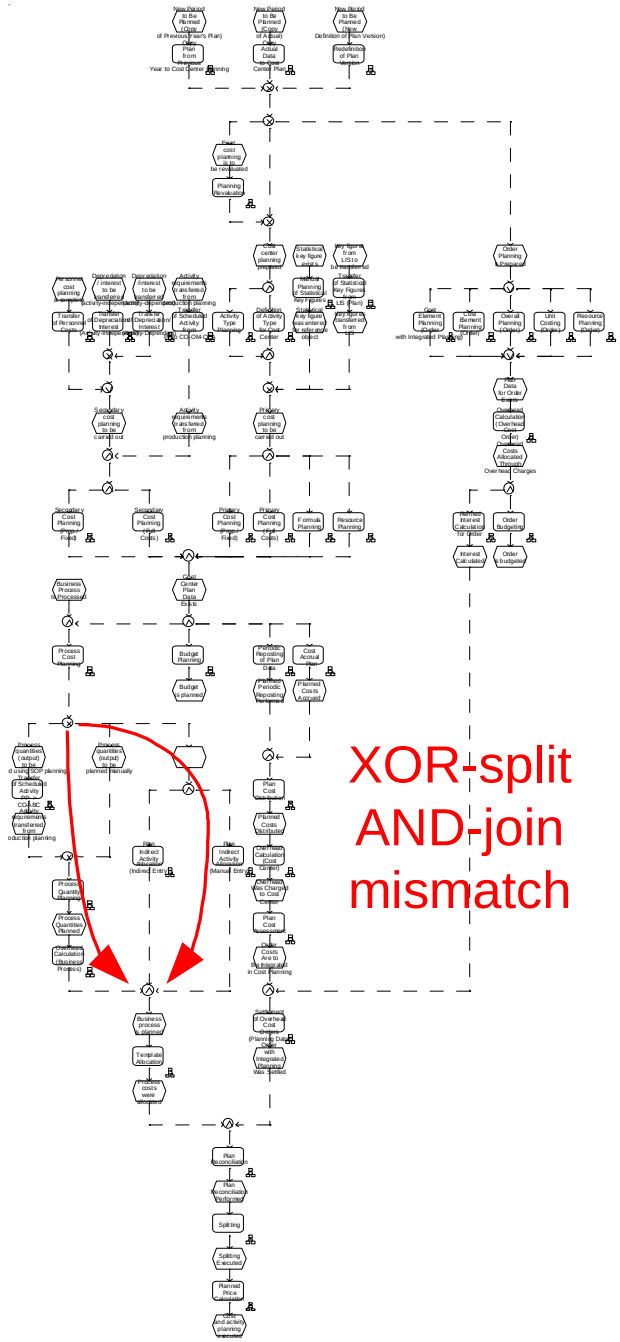


Figure A.71: Revenue and Cost Controlling – Profit and Cost Planning – Cost and Activity Planning (reduced size 7)

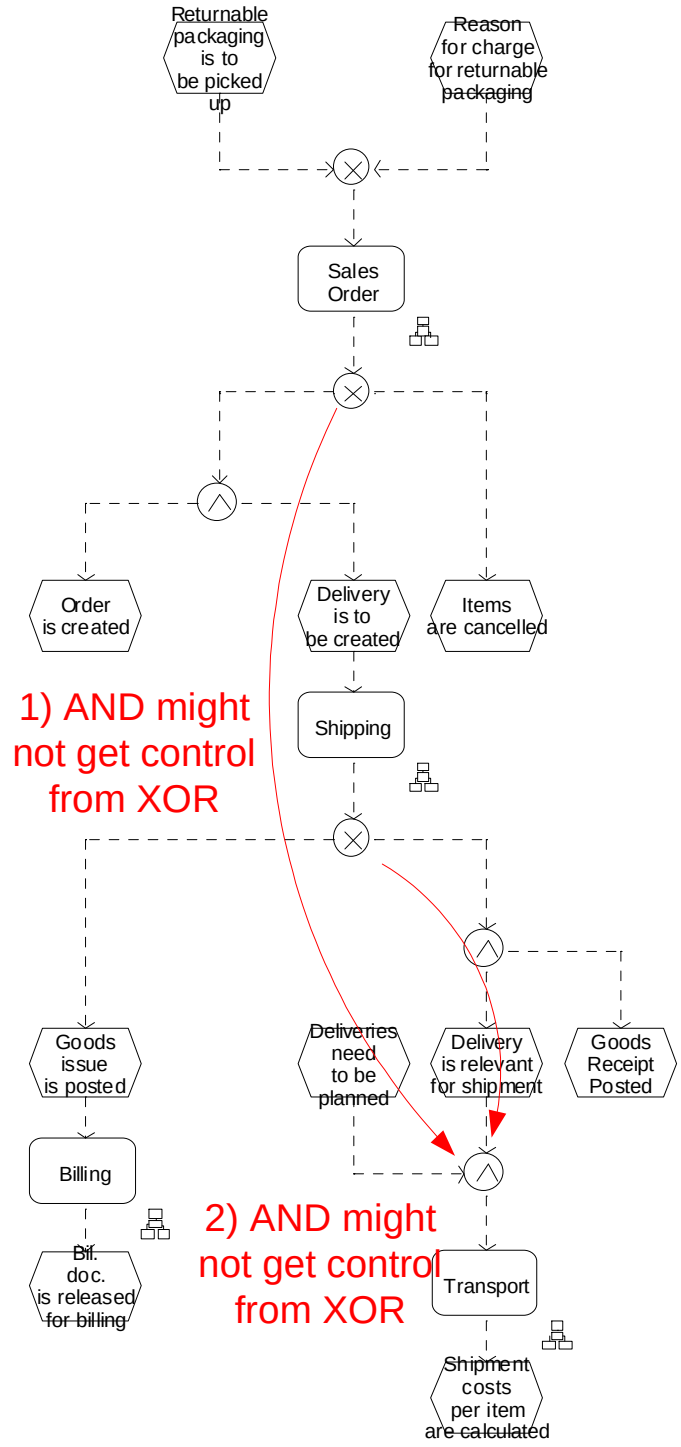


Figure A.72: Sales and Distribution – Empties and Returnable Packaging Handling (completely reduced)

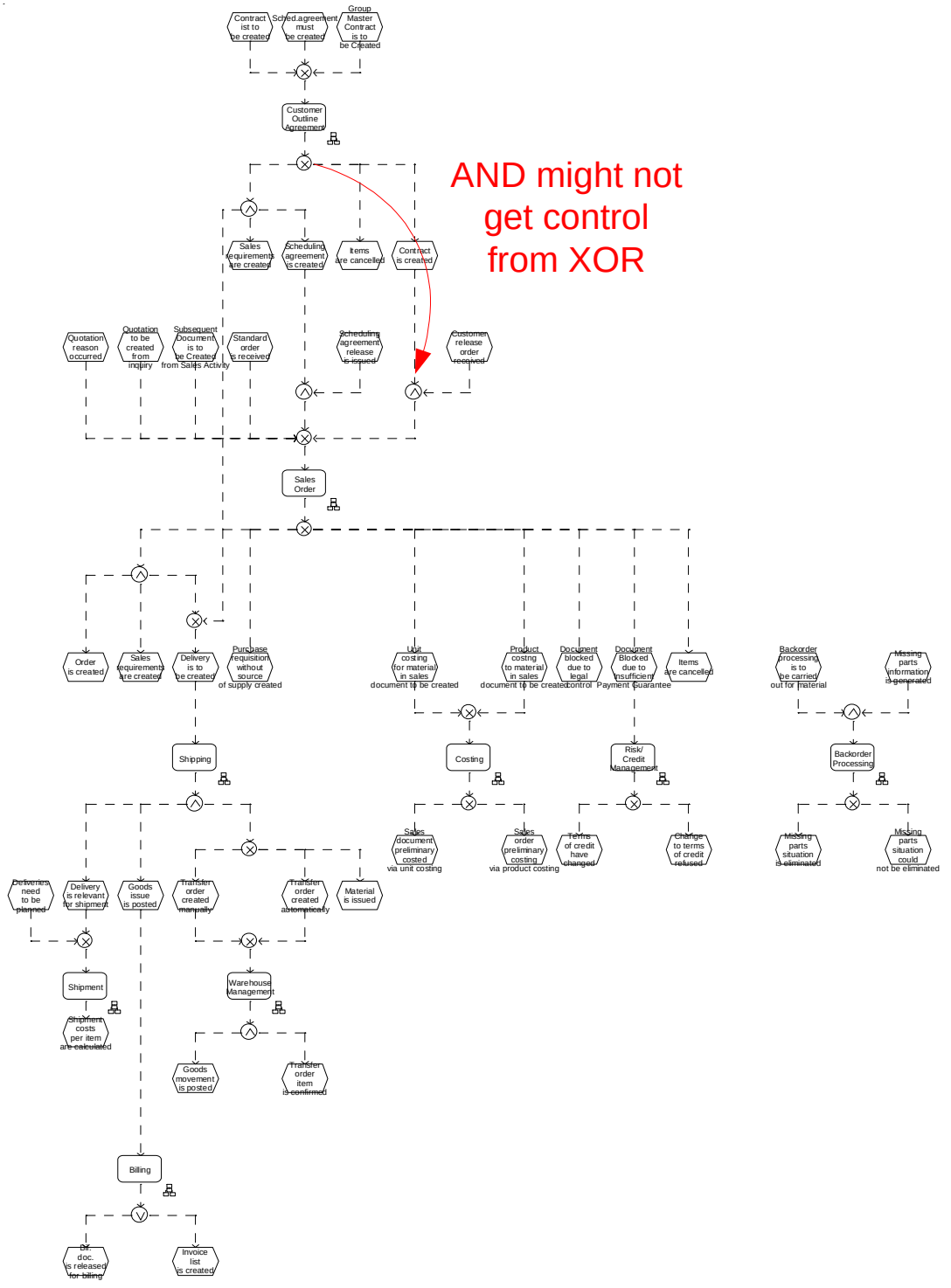


Figure A.73: Sales and Distribution – Sales Order Processing (Standard) (reduced size 14)

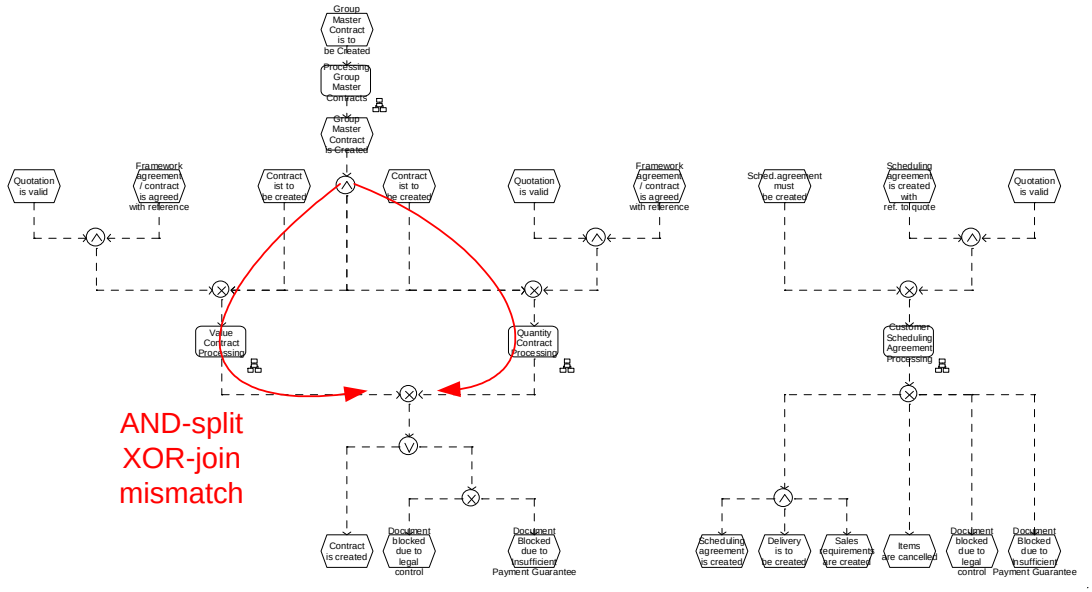


Figure A.74: Sales and Distribution – Sales Order Processing (Standard) – Customer Outline Agreement (completely reduced)

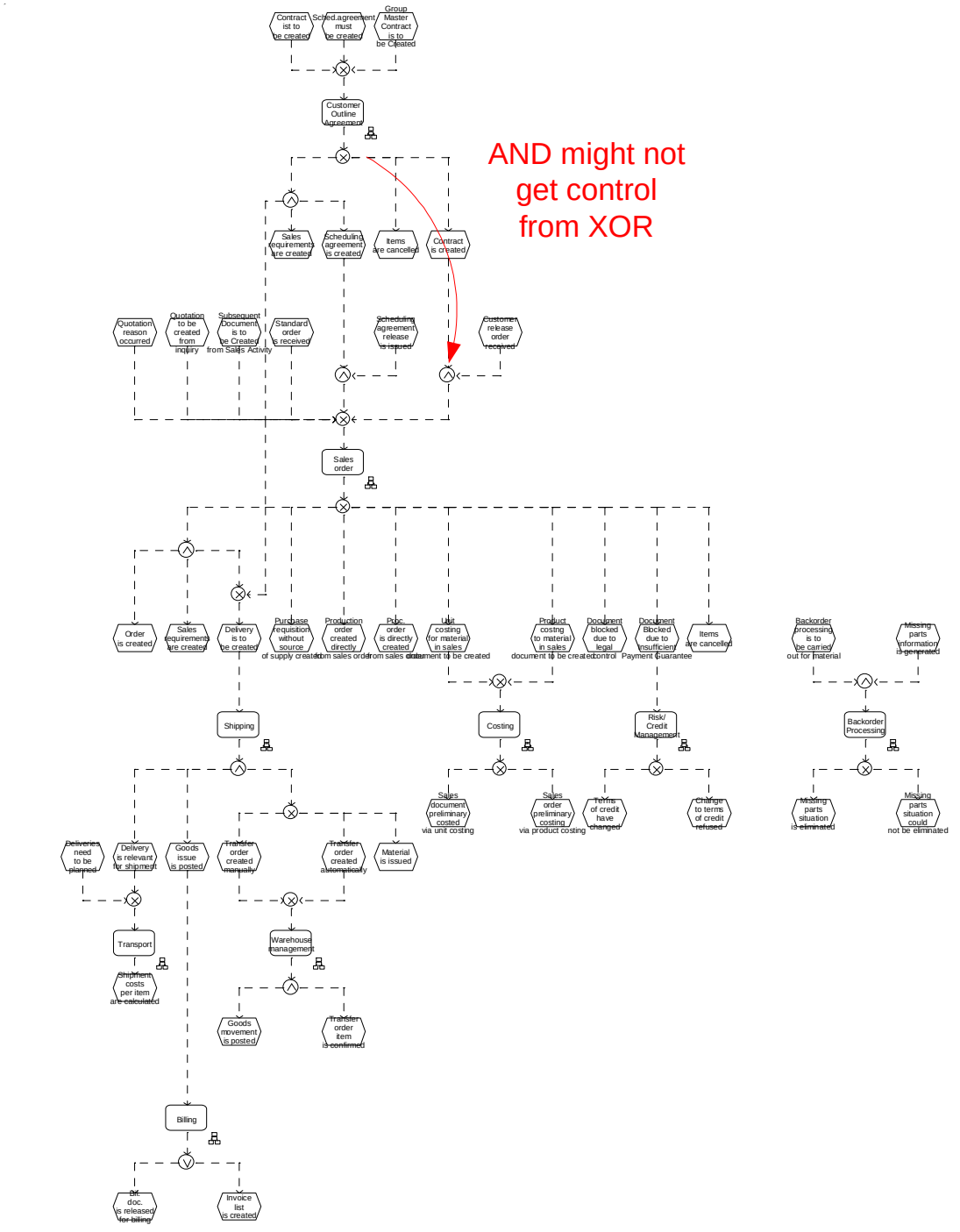


Figure A.75: Sales and Distribution – Sales Order Processing: Make/Assembly To Order (reduced size 14)

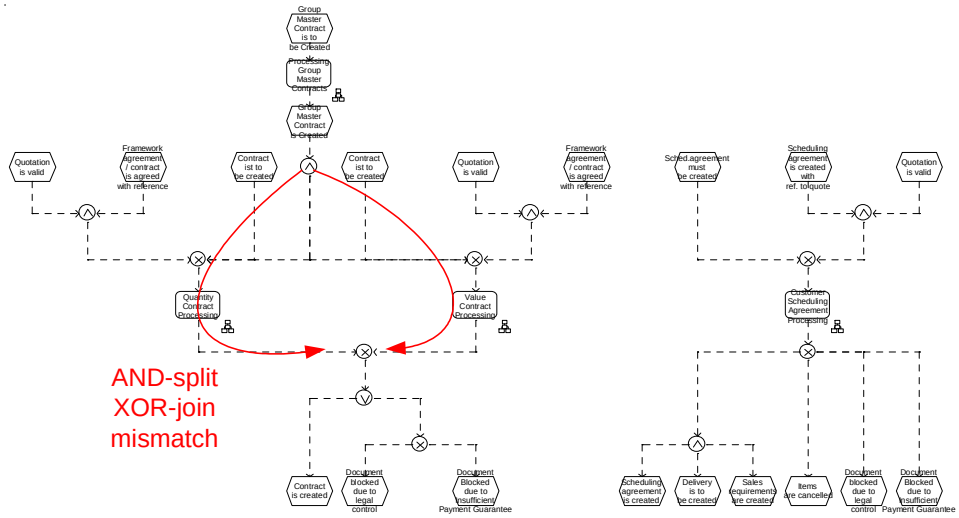


Figure A.76: Sales and Distribution – Sales Order Processing: Make/Assembly To Order – Customer Outline Agreement (completely reduced)

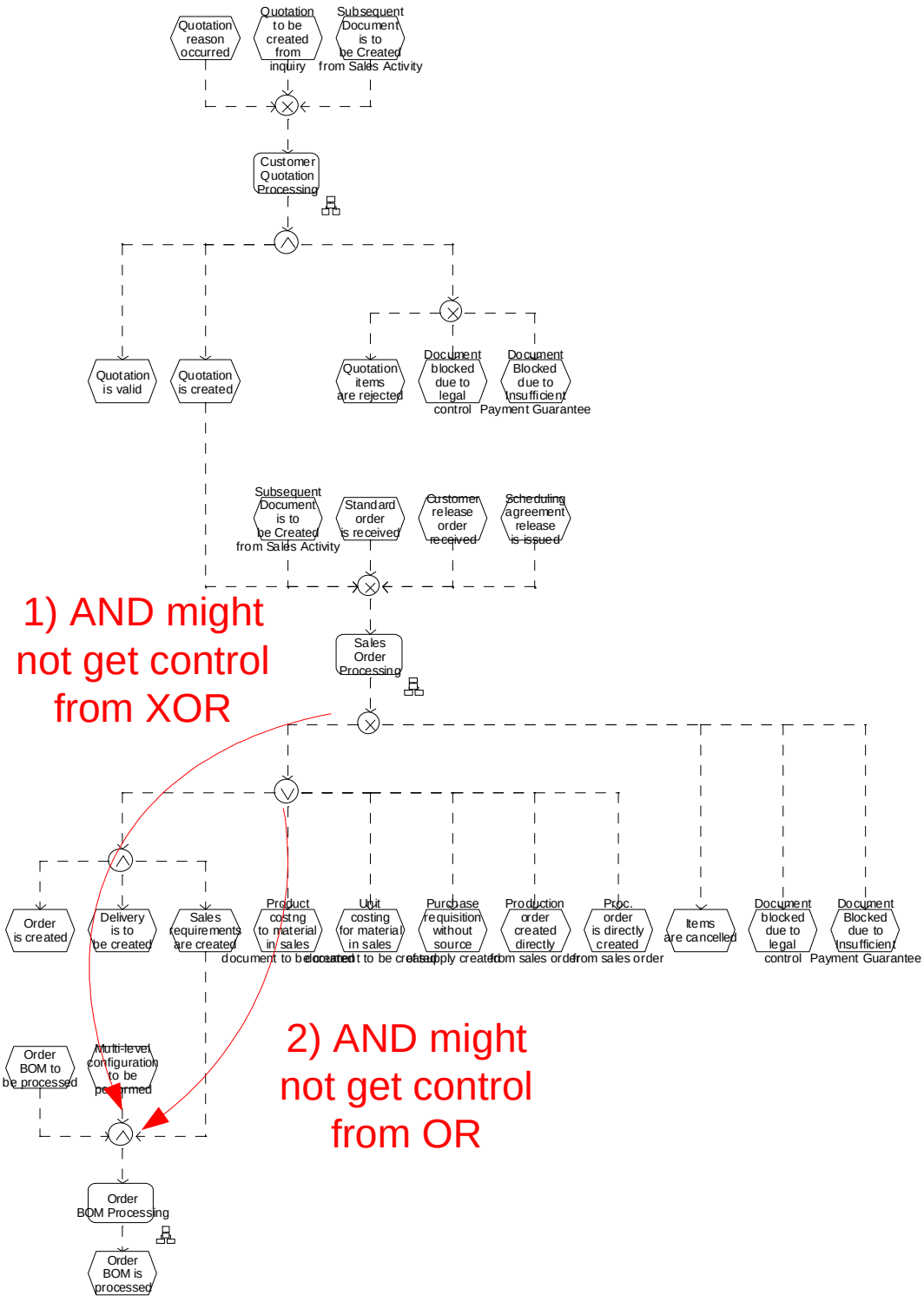


Figure A.77: Sales and Distribution – Sales Order Processing: Make/Assembly To Order – Sales order (completely reduced)

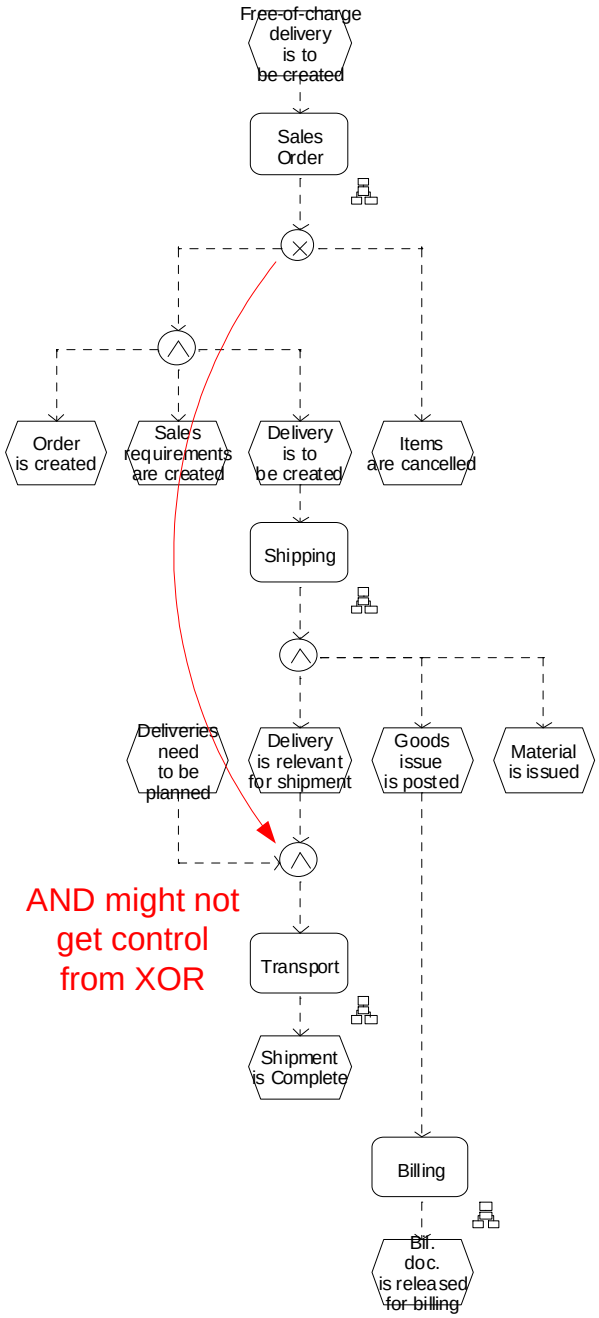


Figure A.78: Sales and Distribution – Sending Samples and Advertising Materials (completely reduced)

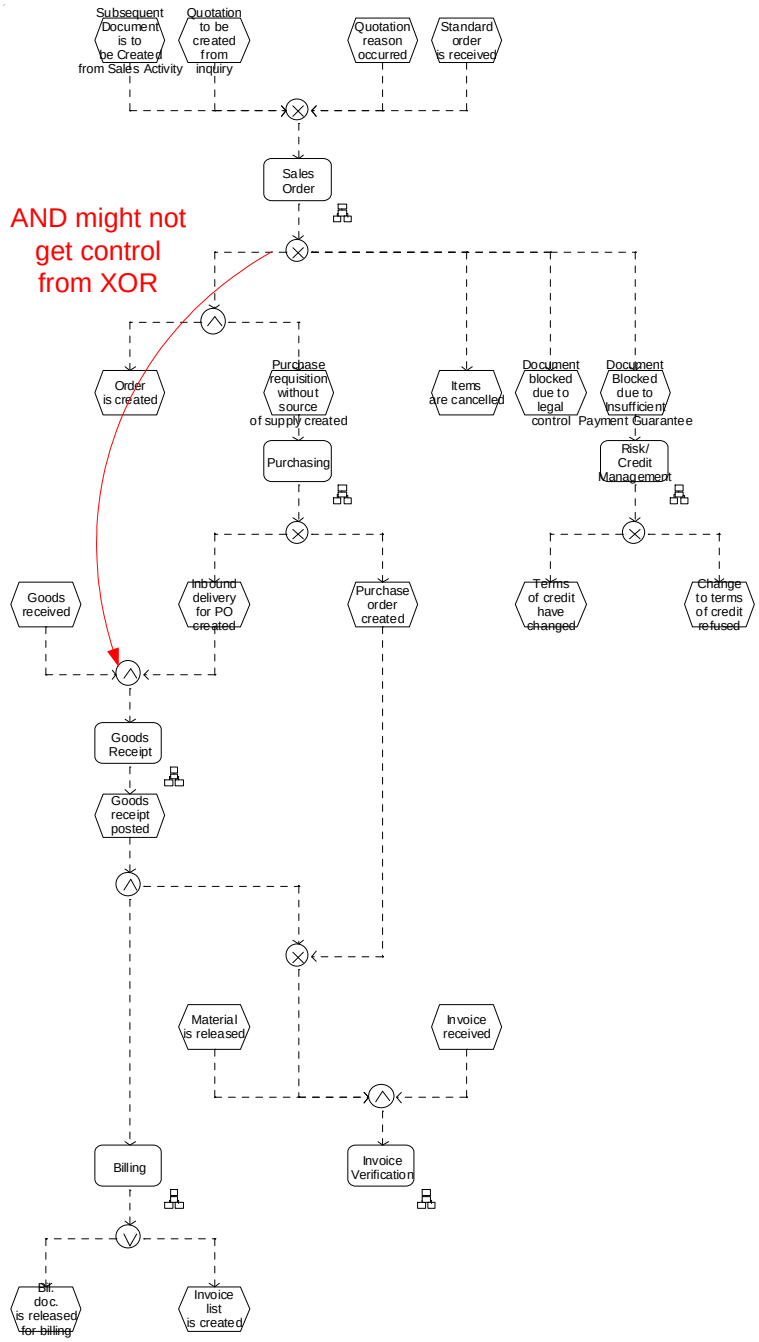


Figure A.79: Sales and Distribution – Third-Party Order Processing (reduced size 8)

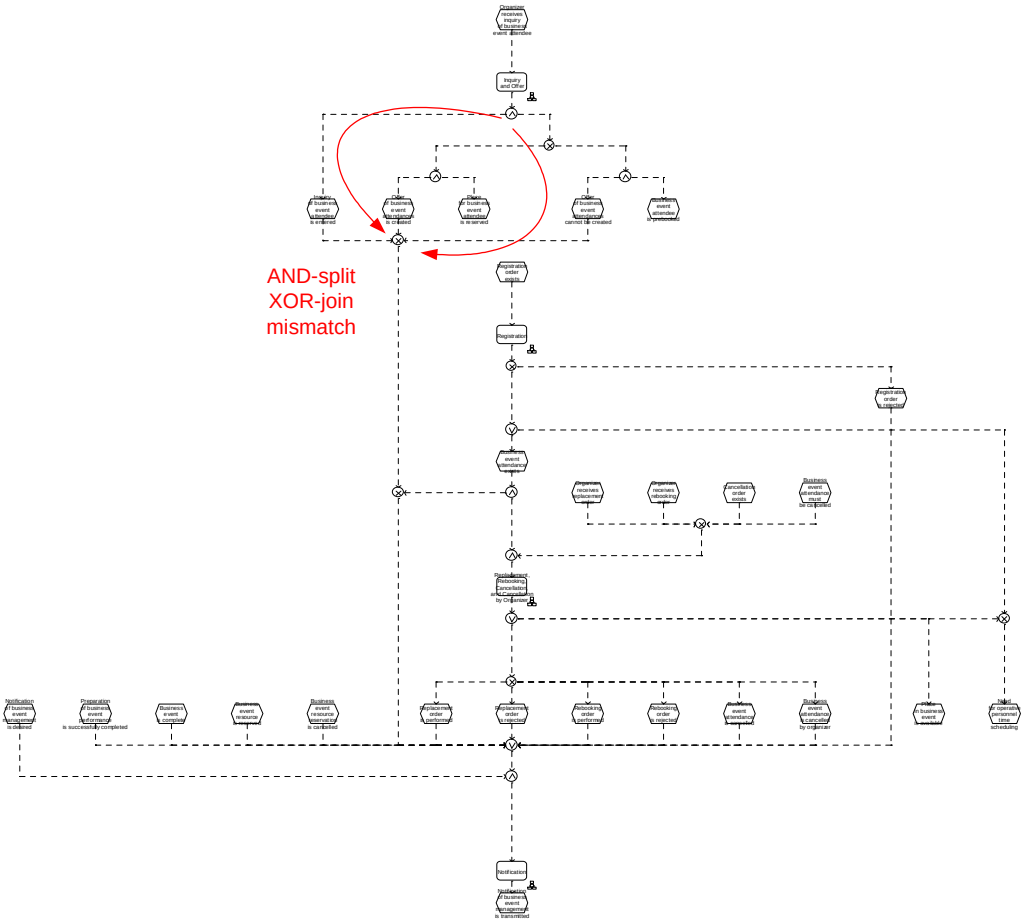


Figure A.80: Training and Event Management – Business Event Attendance Administration (reduced size 17)

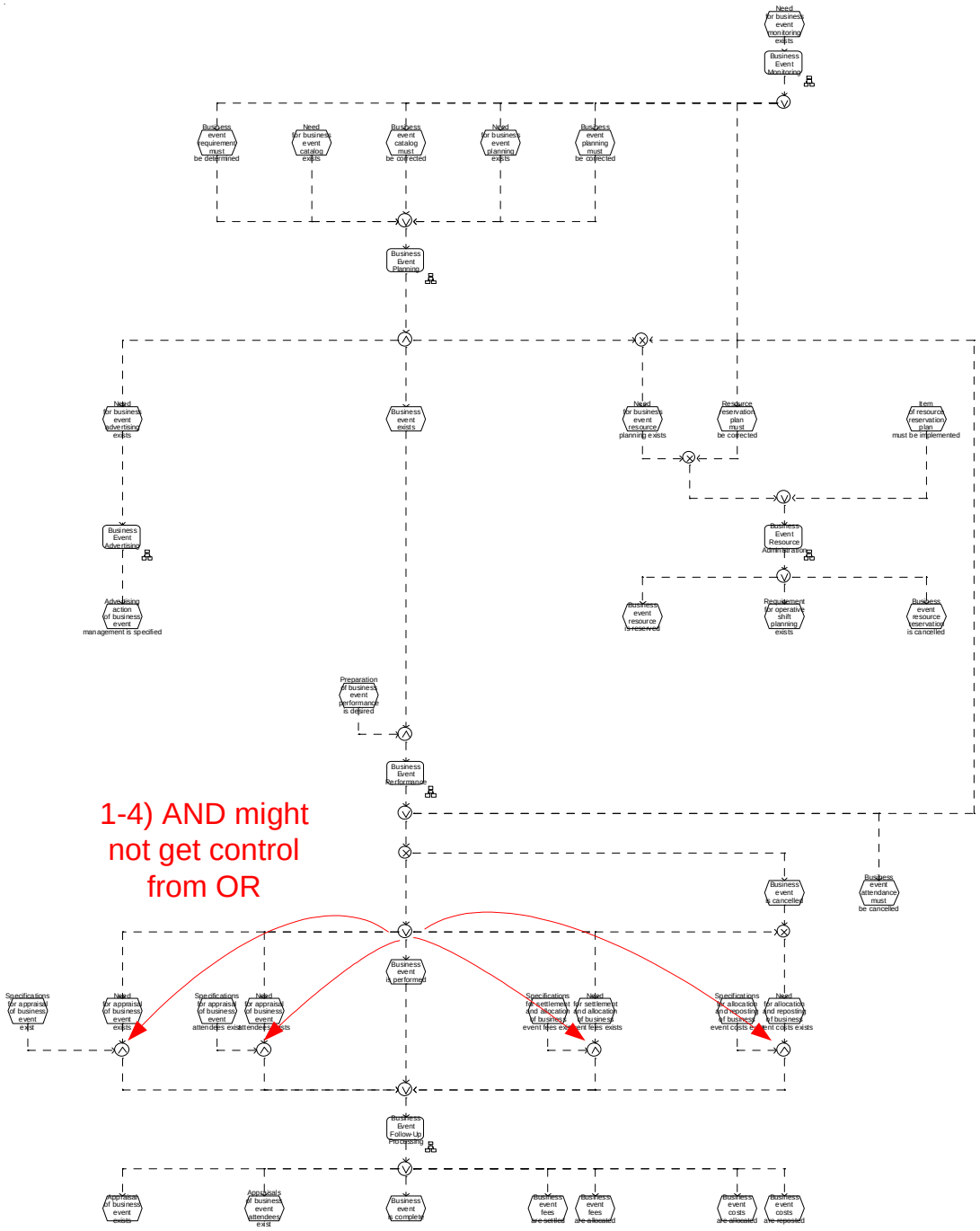


Figure A.81: Training and Event Management – Business Event Planning and Performance (reduced size 22)

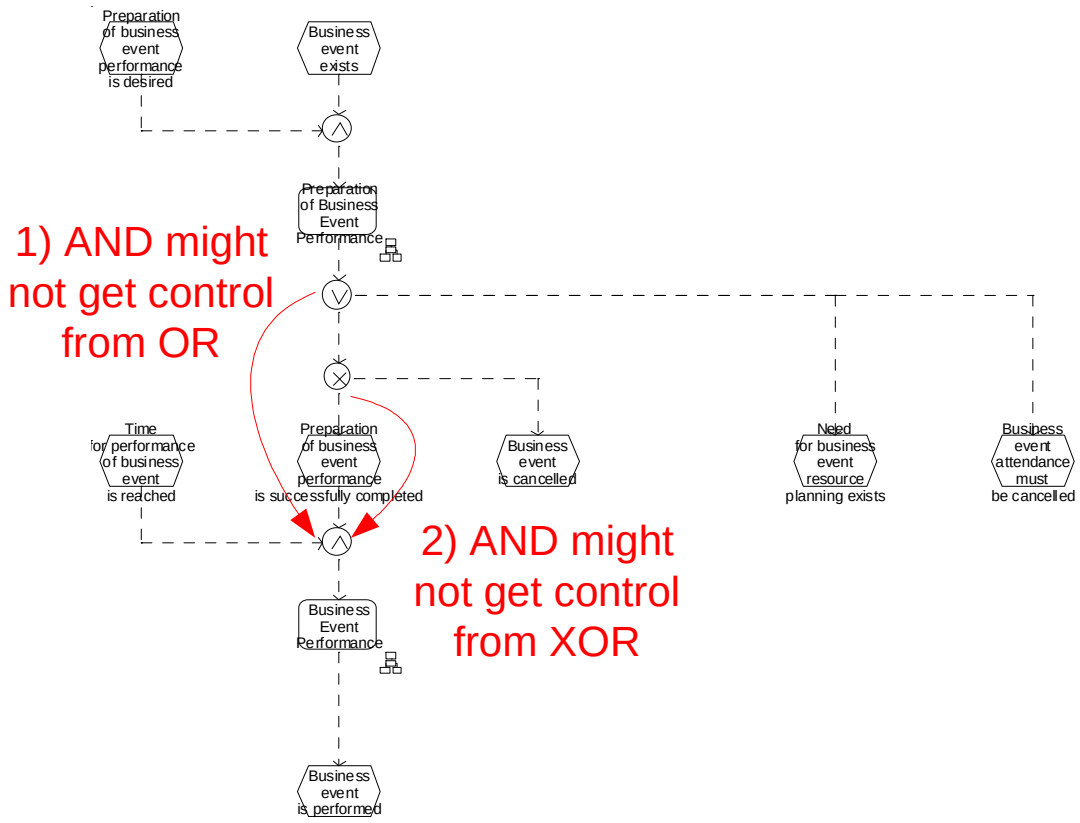


Figure A.82: Training and Event Management – Business Event Planning and Performance – Business Event Performance (completely reduced)

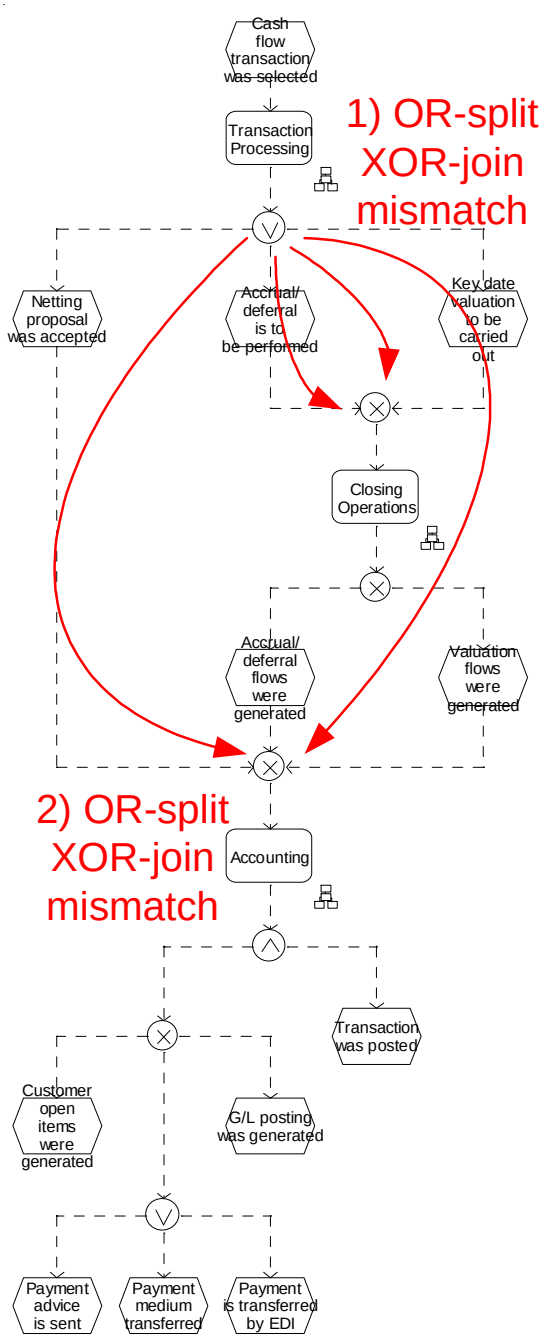


Figure A.83: Treasury – Cash Flow Transactions (TR-MM) (completely reduced)

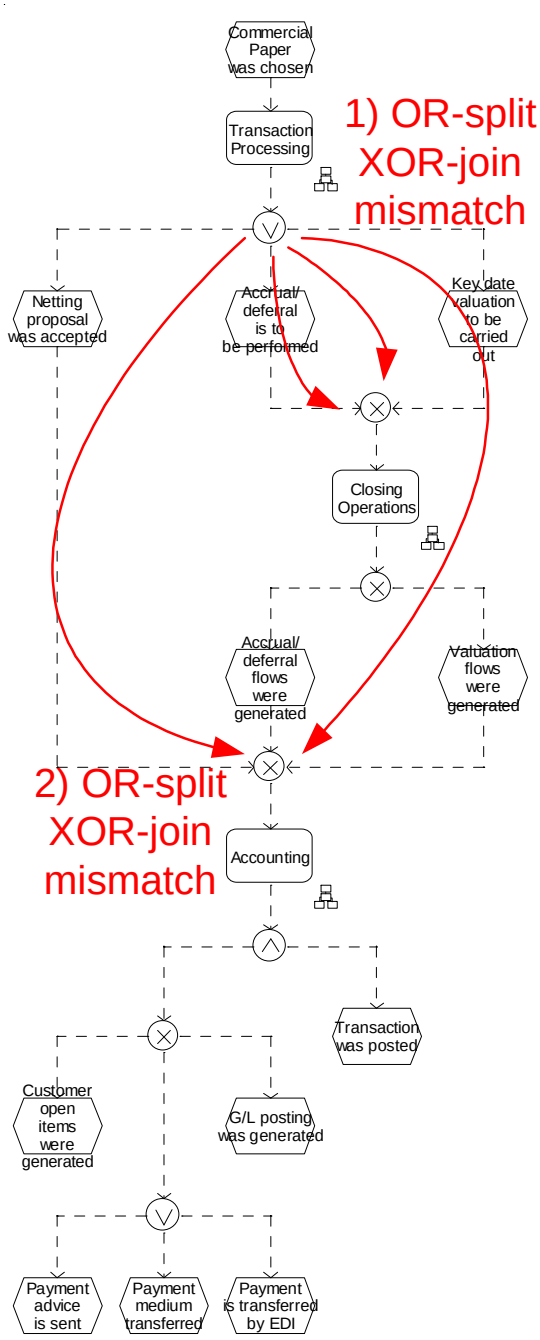


Figure A.84: Treasury – Commercial Paper (TR-MM) (completely reduced)

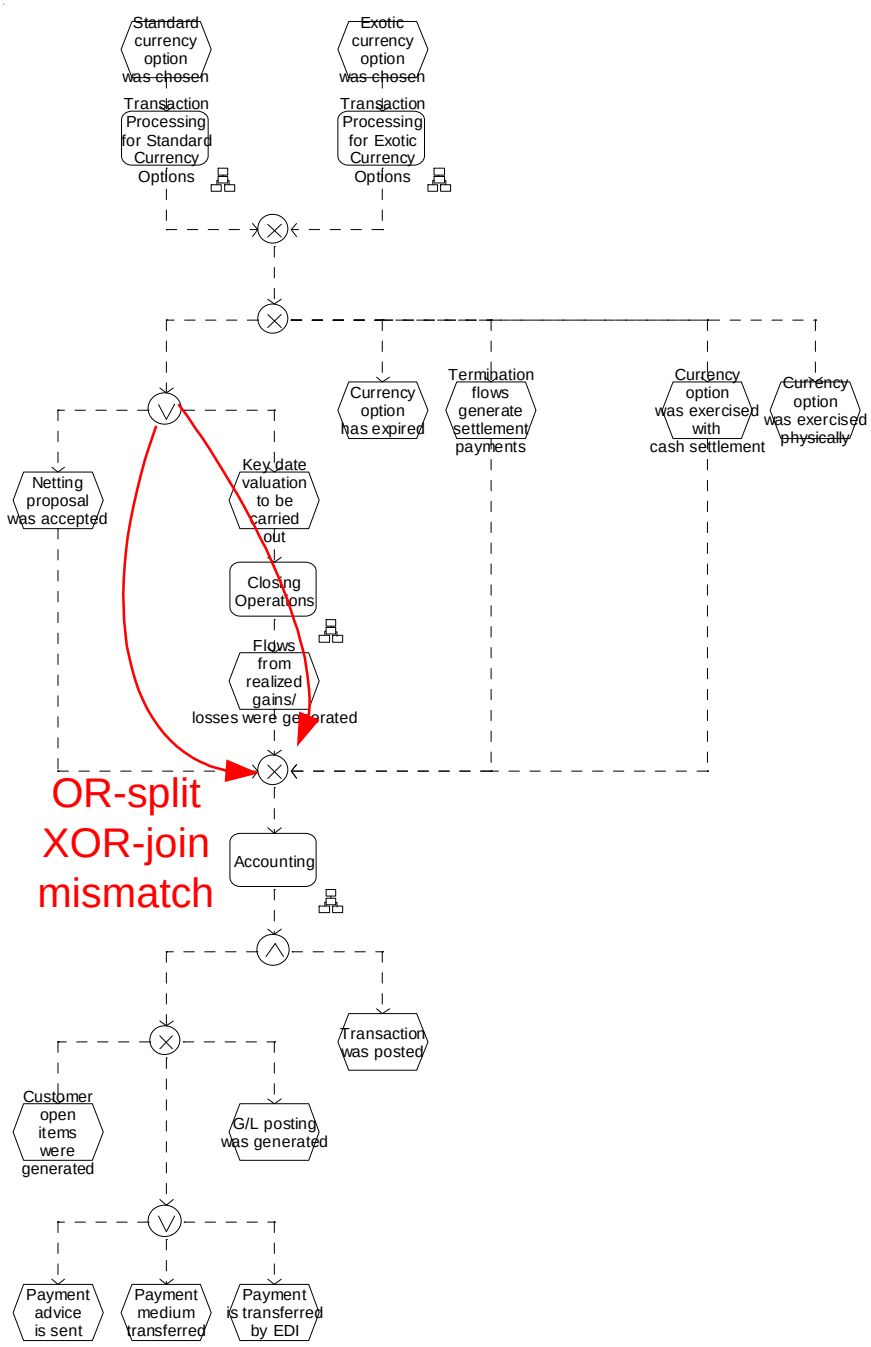


Figure A.85: Treasury – Currency Options (TR-FX) (completely reduced)

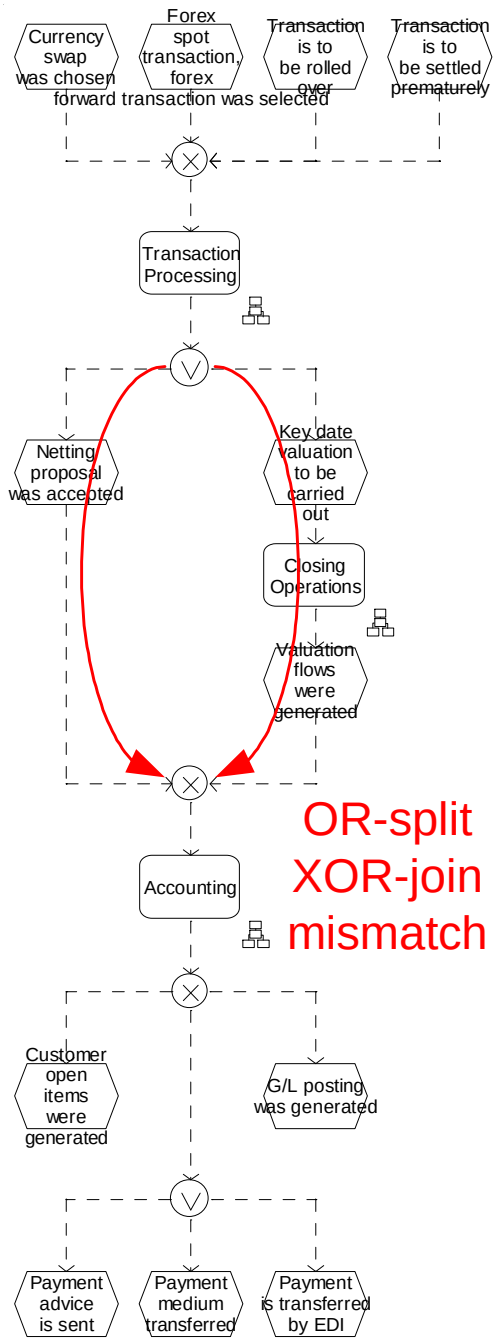


Figure A.86: Treasury – Forex Spot, Forward and Swap Transactions (TR-FX) (completely reduced)

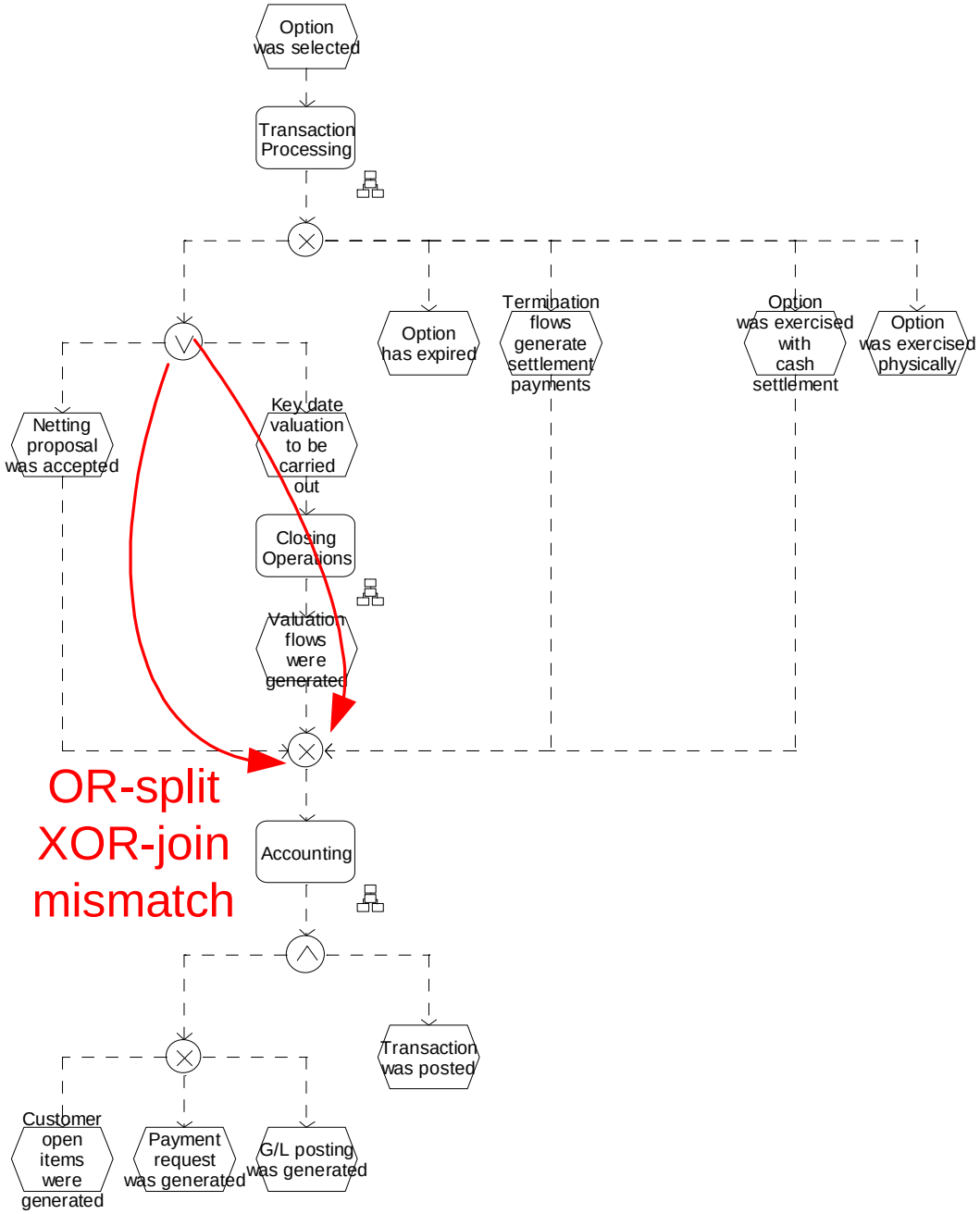


Figure A.87: Treasury – Options on Interest Rate Instruments and Securities (TR-DE) (completely reduced)

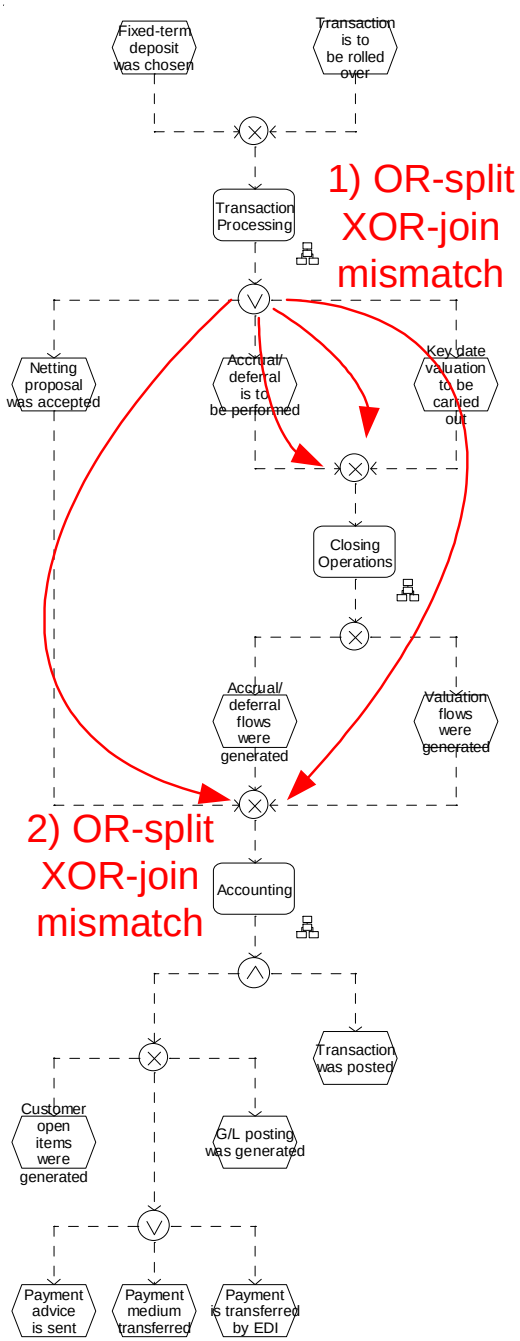


Figure A.88: Treasury – Process Fixed-Term Deposit (TR-MM) (completely reduced)

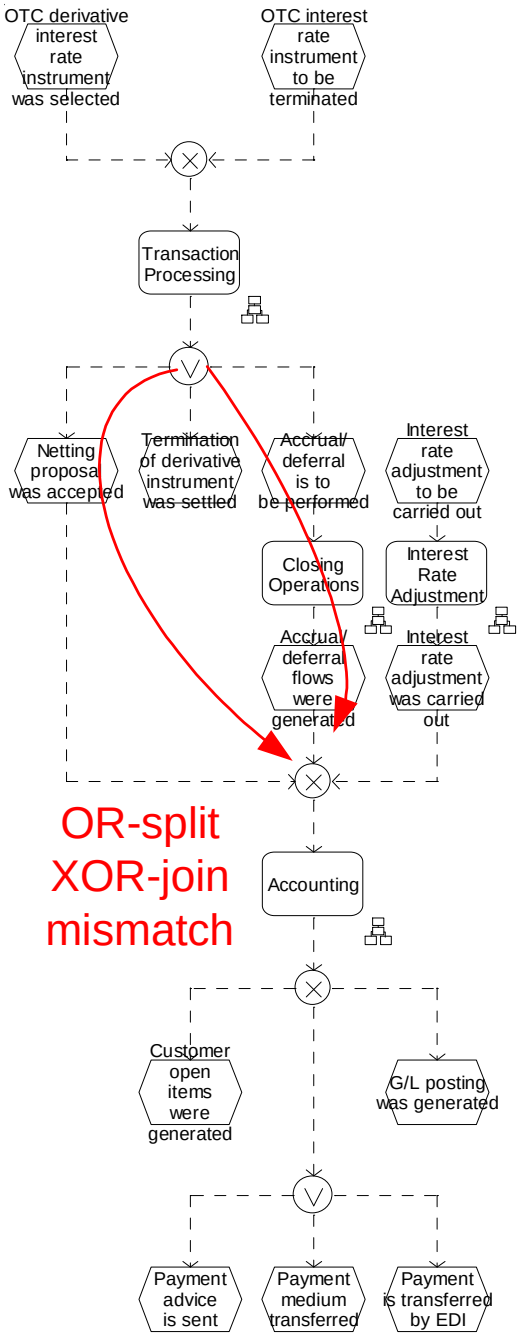


Figure A.89: Treasury – Process OTC Derivative Transactions (TR-DE) (reduced size 6)

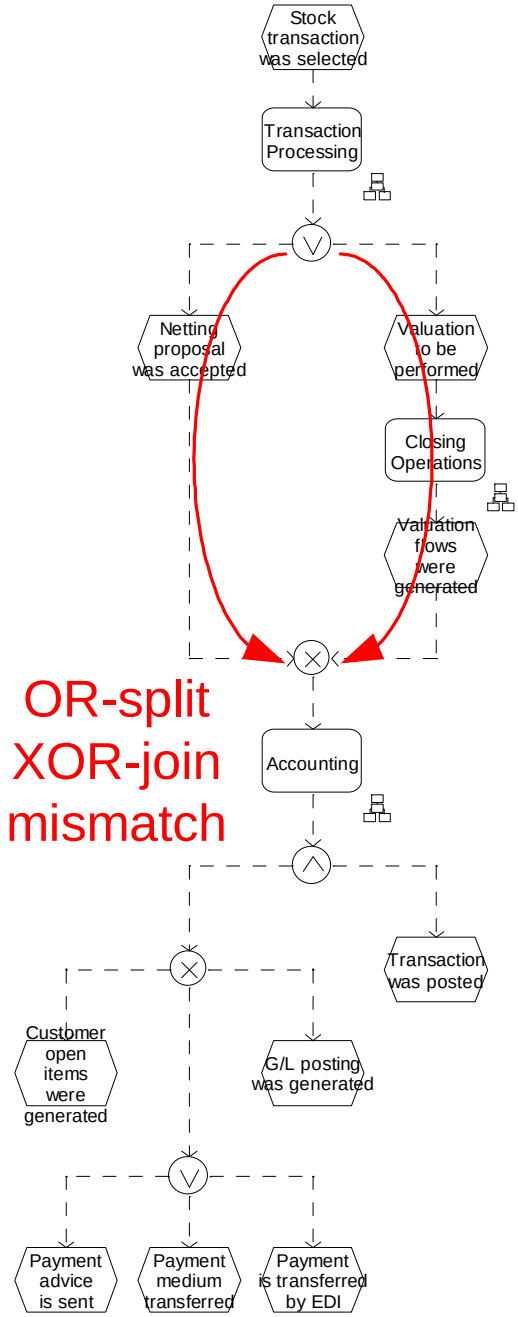


Figure A.90: Treasury – Stocks (TR-SE) (completely reduced)

Appendix B

EPCs not completely reduced

This appendix shows those EPCs of the SAP Reference Model that were not completely reduced and for which *xoEPC* did not find an error. We give the rest size in brackets and indicate whether ProM identified them to be sound or unsound.

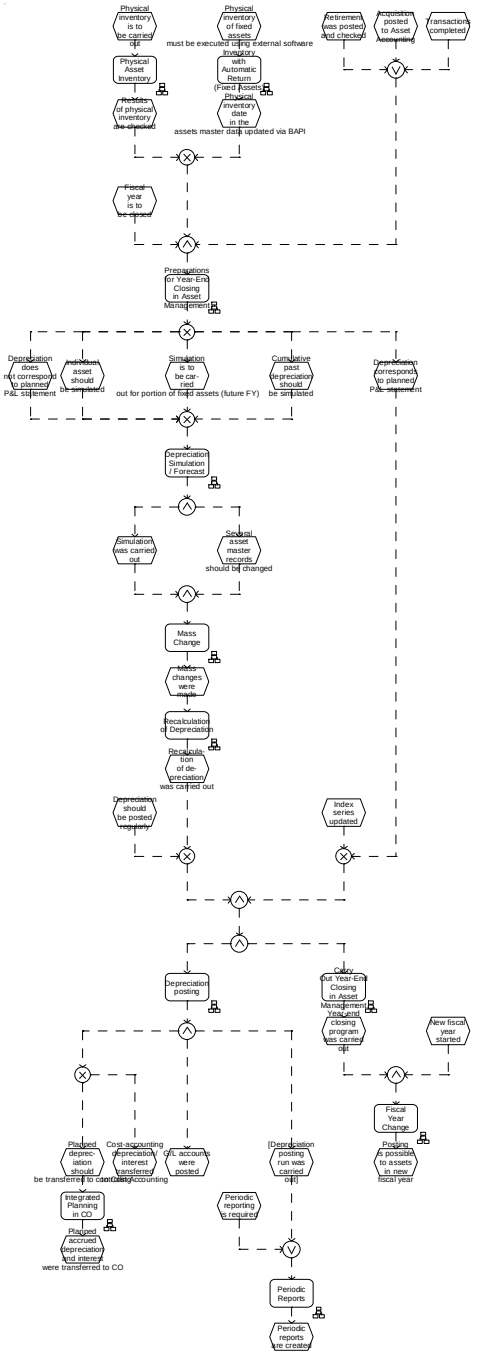


Figure B.1: Asset Accounting – Handling Fixed Assets – Closing Operations (Asset Accounting) (reduced size 14, unsound)

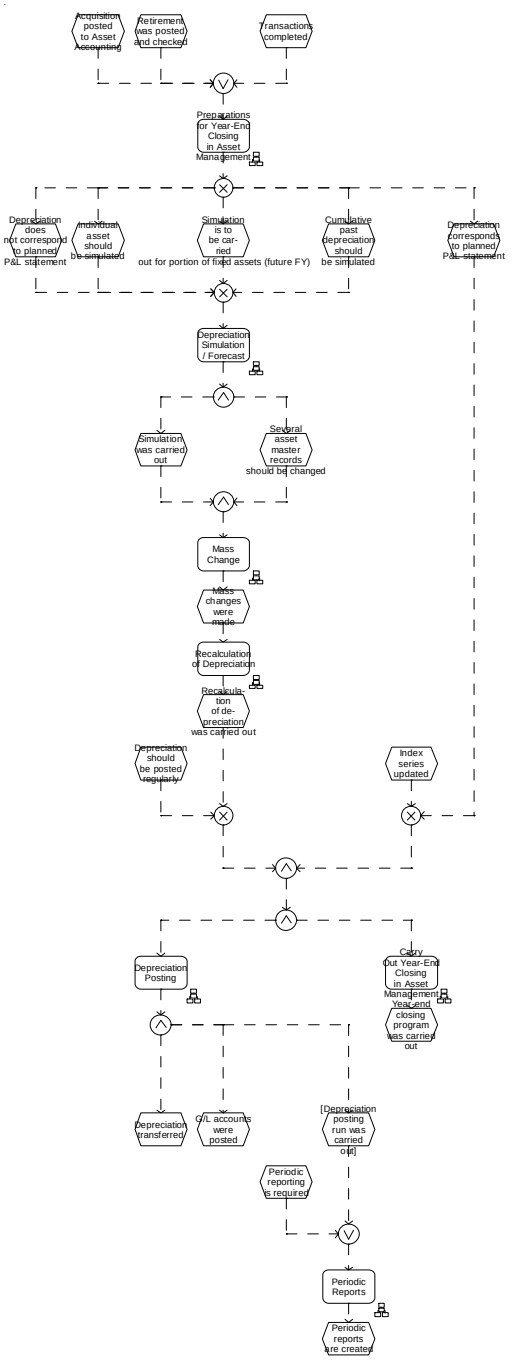


Figure B.2: Asset Accounting – Handling of Leased Assets – Closing Operations (reduced size 10, unsound)

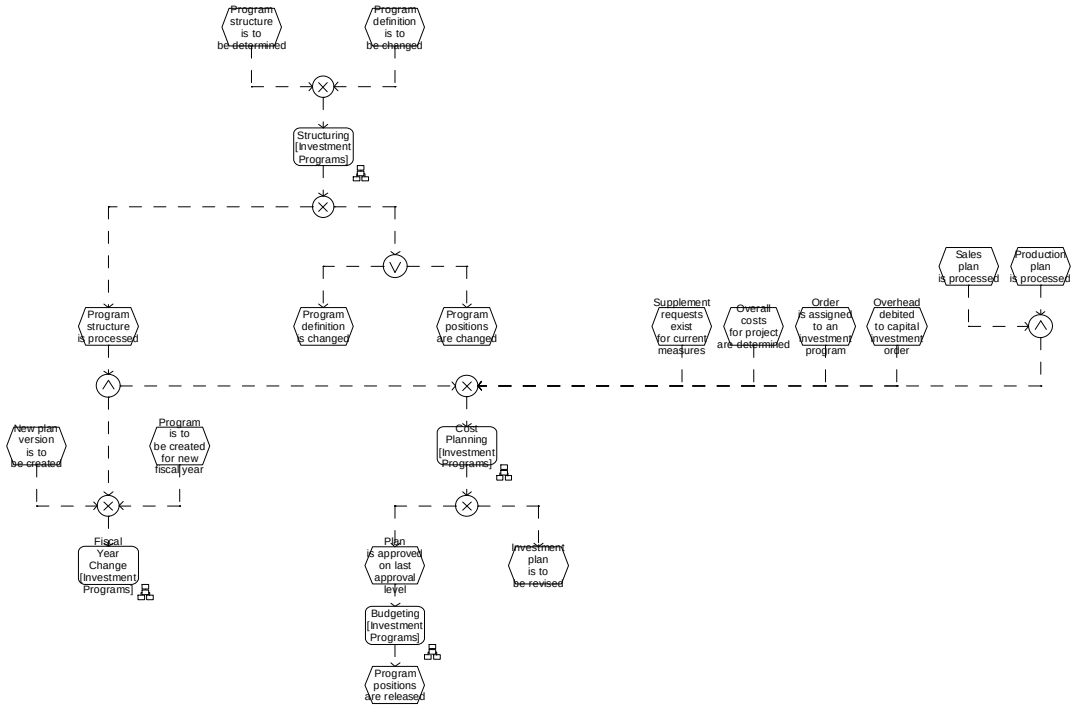


Figure B.3: Asset Accounting – Investment Program Handling (Capital Investments) (reduced size 10, sound)

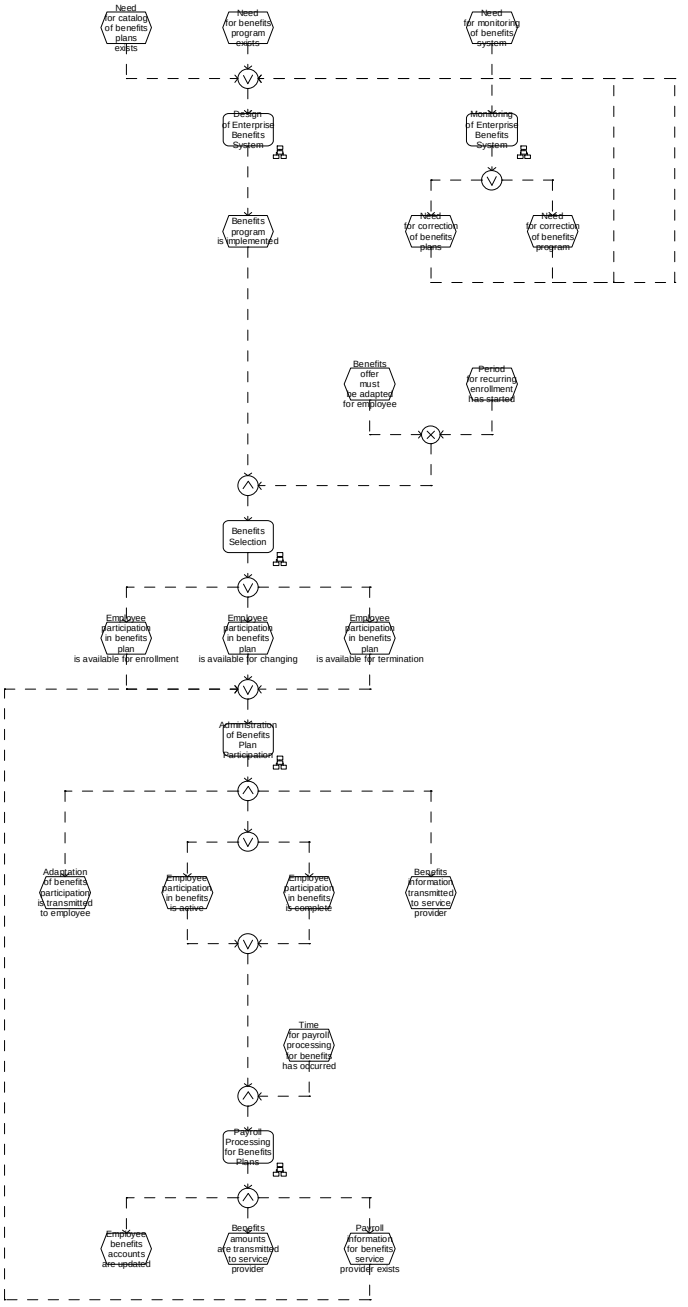


Figure B.4: Benefits Administration – Benefits Administration (reduced size 8, unsound)

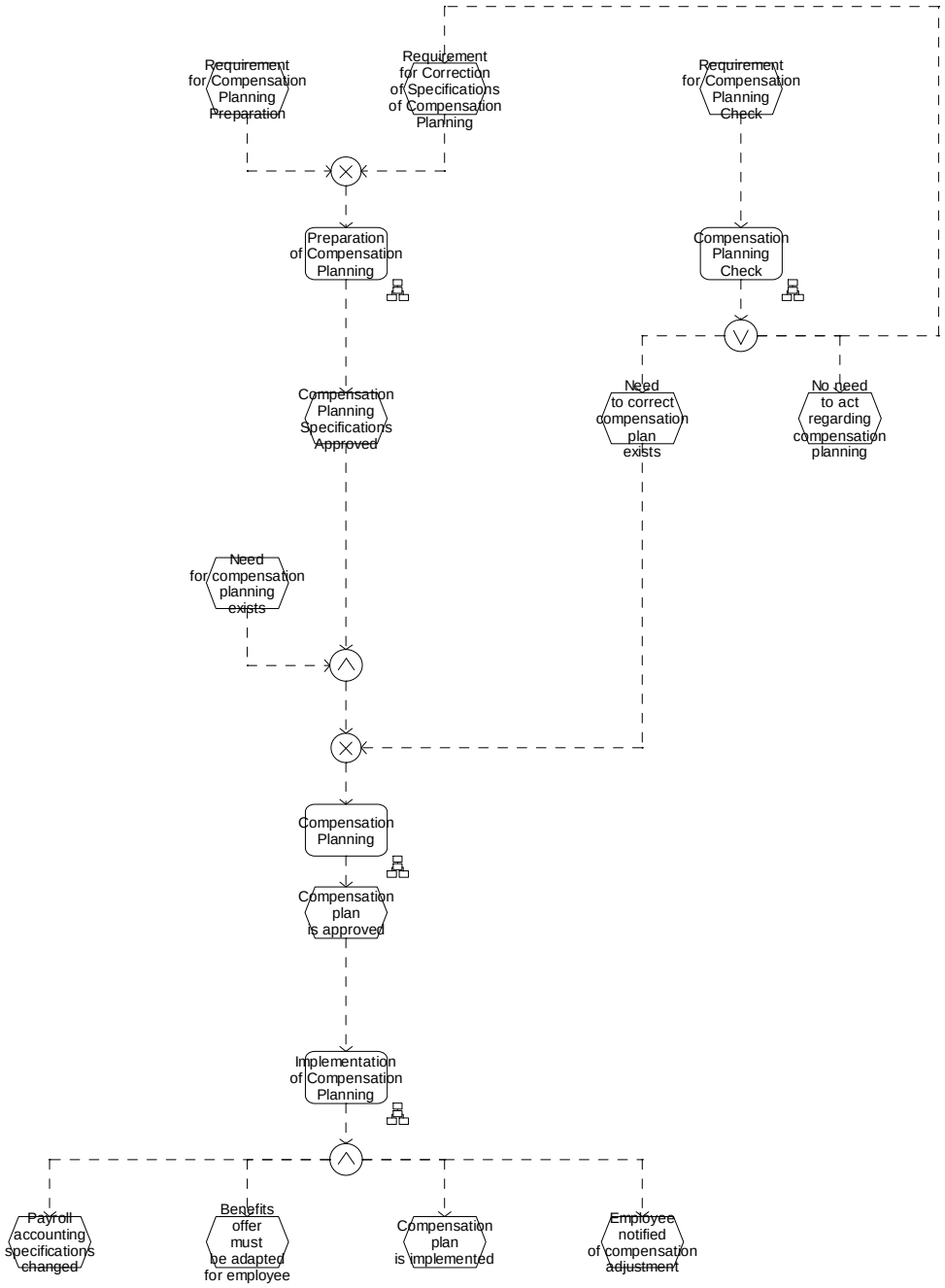


Figure B.5: Compensation Management – Compensation Planning (reduced size 9, unsound)

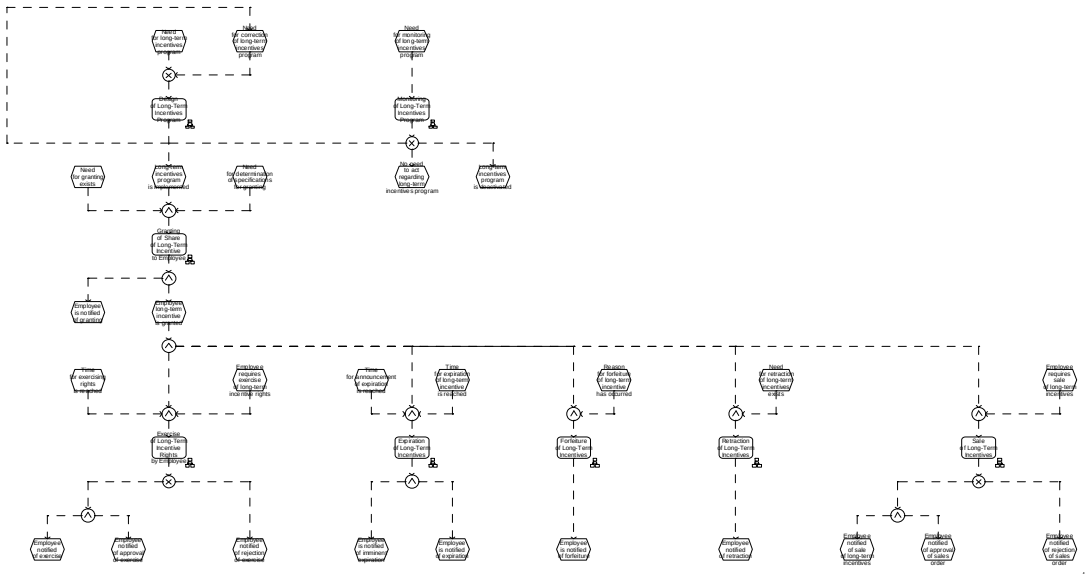


Figure B.6: Compensation Management – Long-Term Incentives (reduced size 23, unsound)

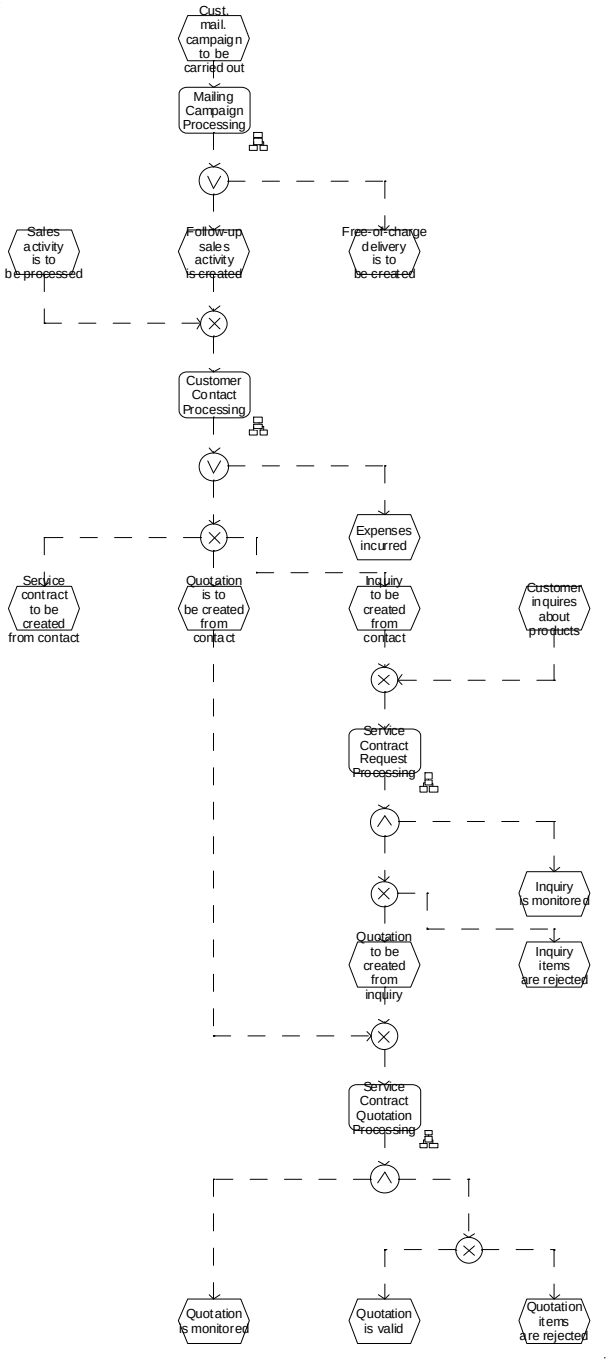


Figure B.7: Customer Service – Long-Term Service Agreements – Presales Activities (reduced size 15, sound)

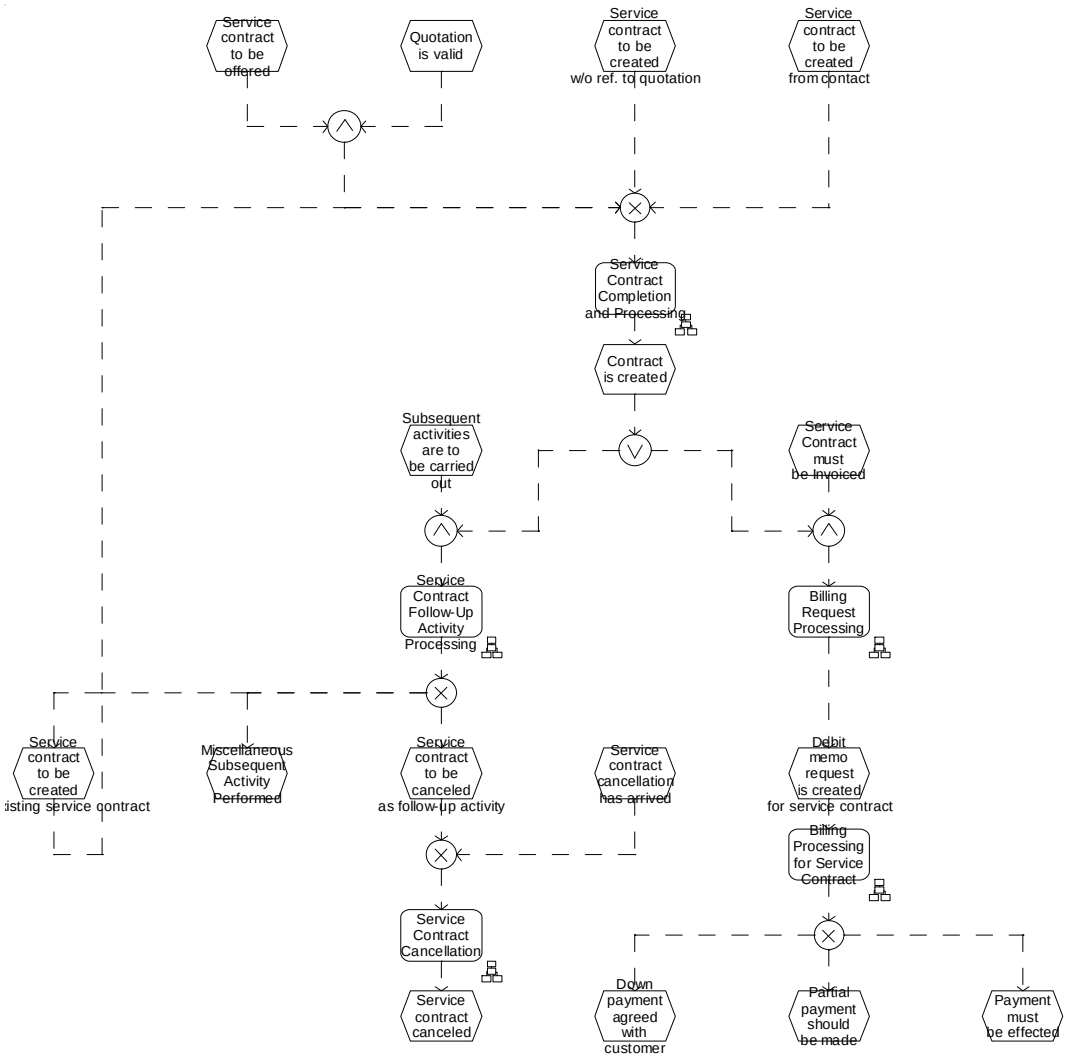


Figure B.8: Customer Service – Long-Term Service Agreements – Service Contract Processing (reduced size 13, unsound)

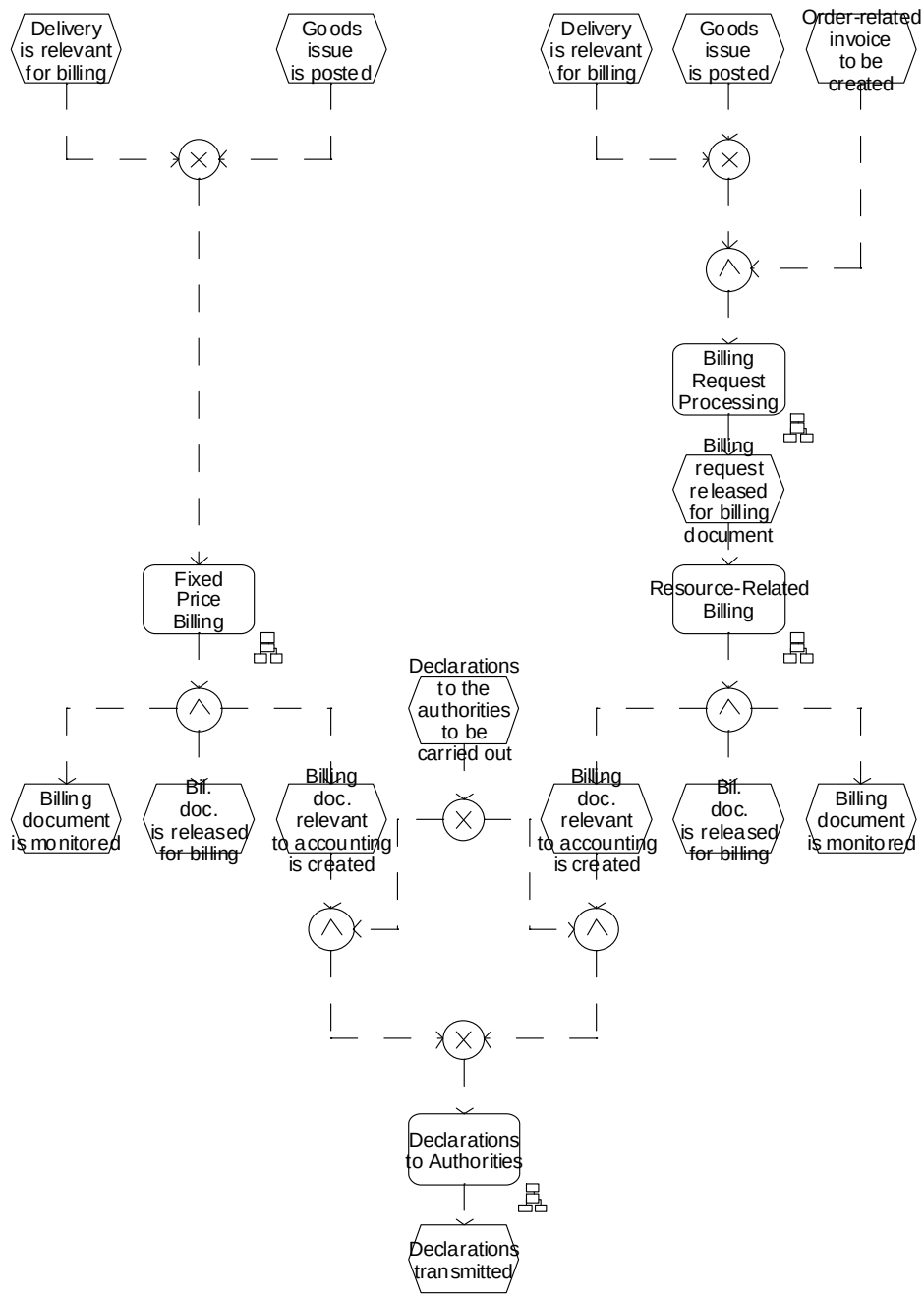


Figure B.9: Customer Service – Repairs Processing at Customer (Field Service) – Billing (reduced size 8, unsound)

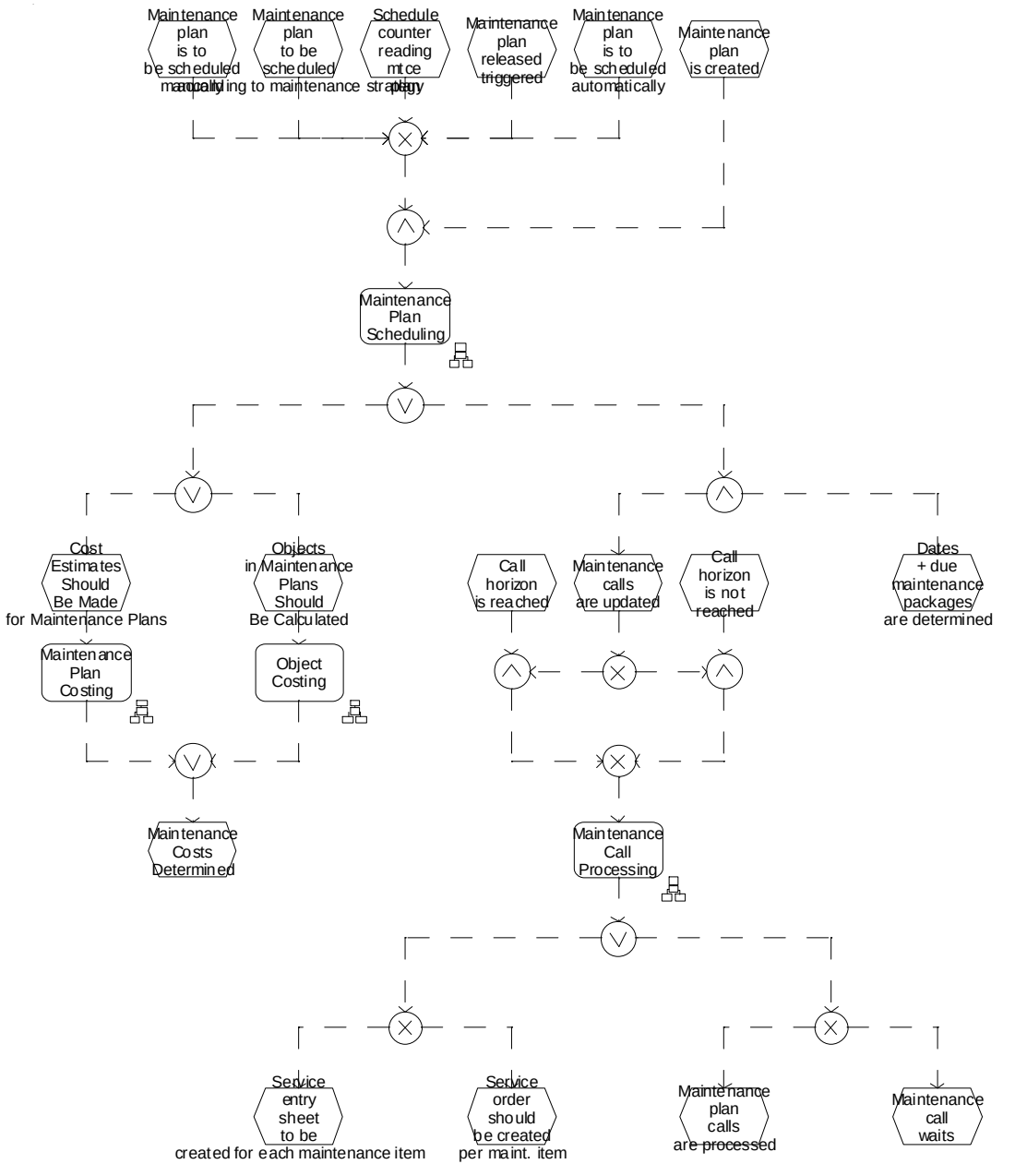


Figure B.10: Customer Service – Repairs Processing at Customer (Field Service) – Maintenance Planning (reduced size 10, unsound)

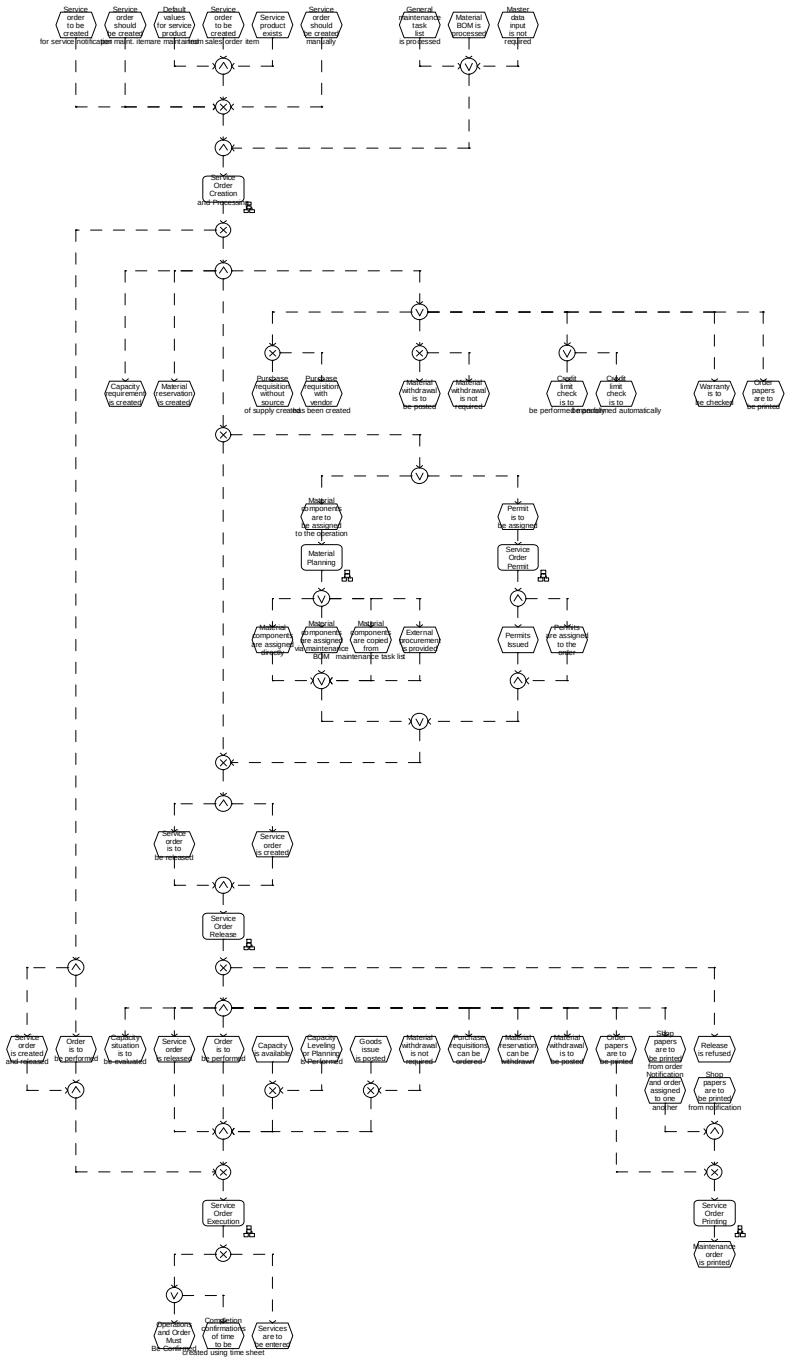


Figure B.11: Customer Service – Repairs Processing at Customer (Field Service) – Service Order (reduced size 11, unsound)

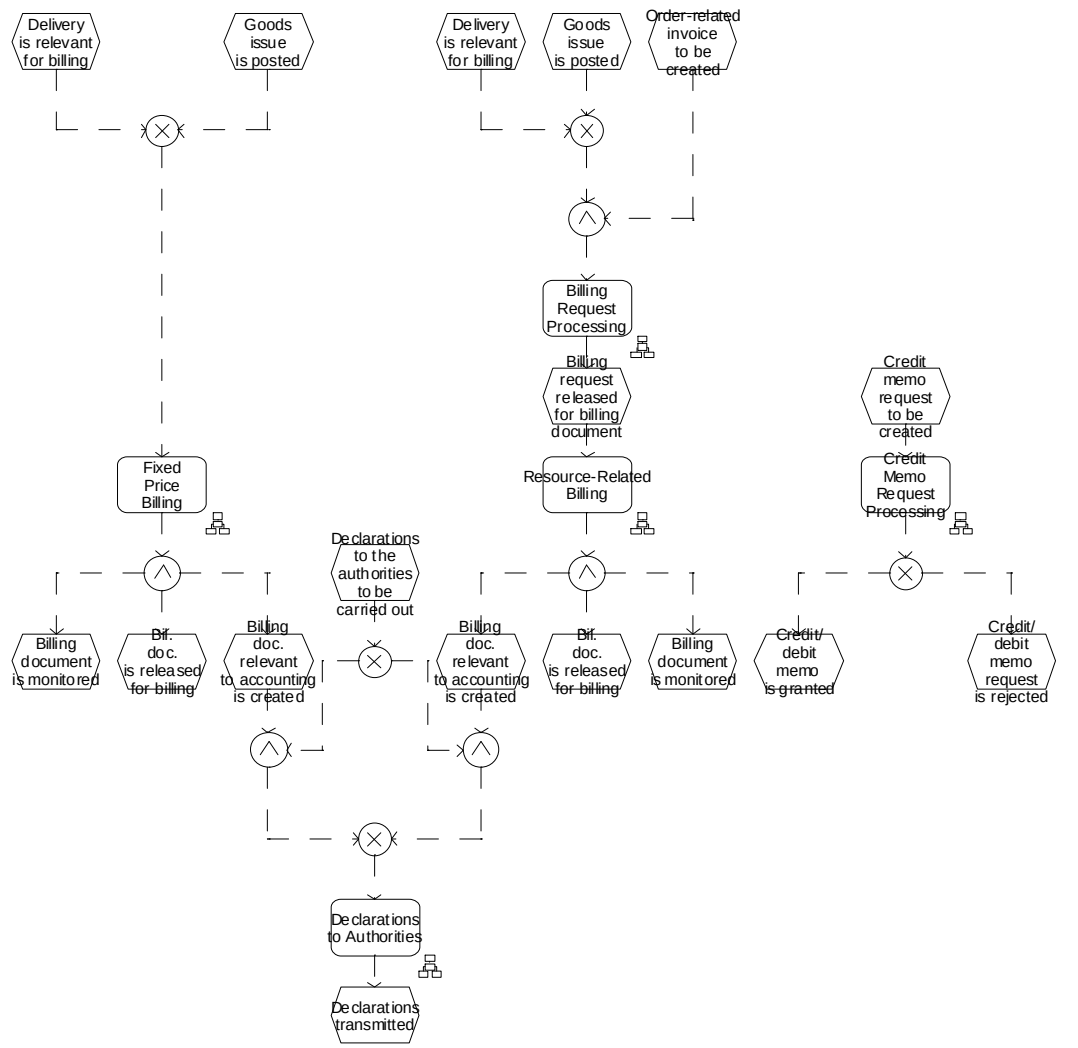


Figure B.12: Customer Service – Repairs Processing in Service Center (Inhouse) – Billing (reduced size 8, unsound)

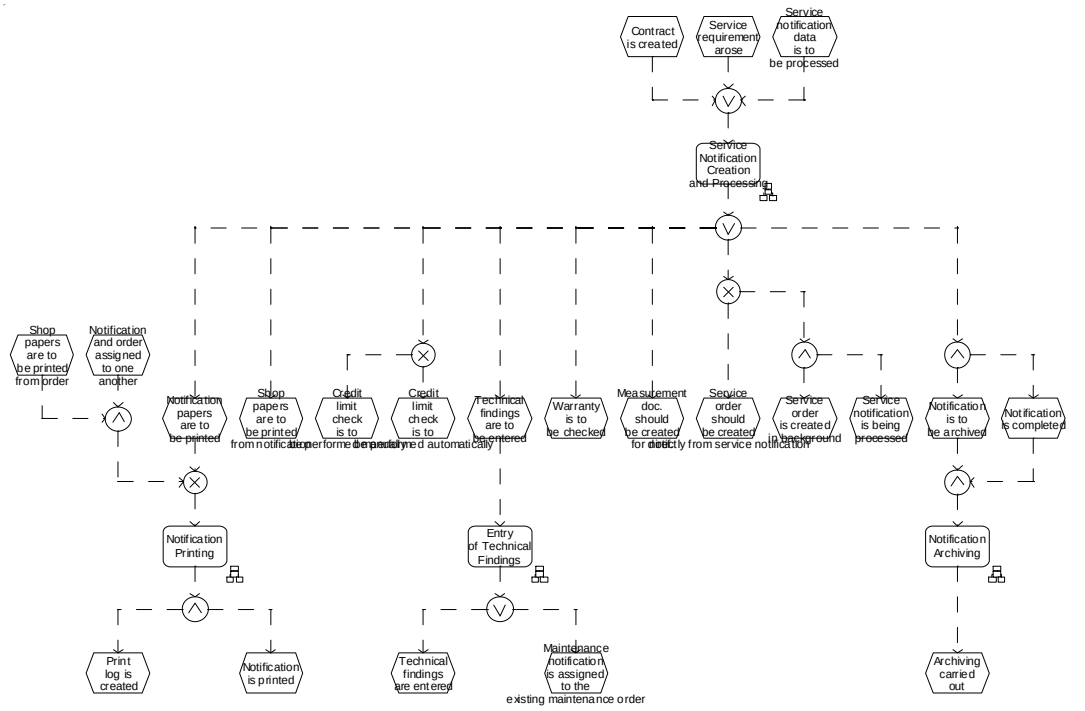


Figure B.13: Customer Service – Repairs Processing in Service Center (Inhouse) – Service Notification (reduced size 6, sound)

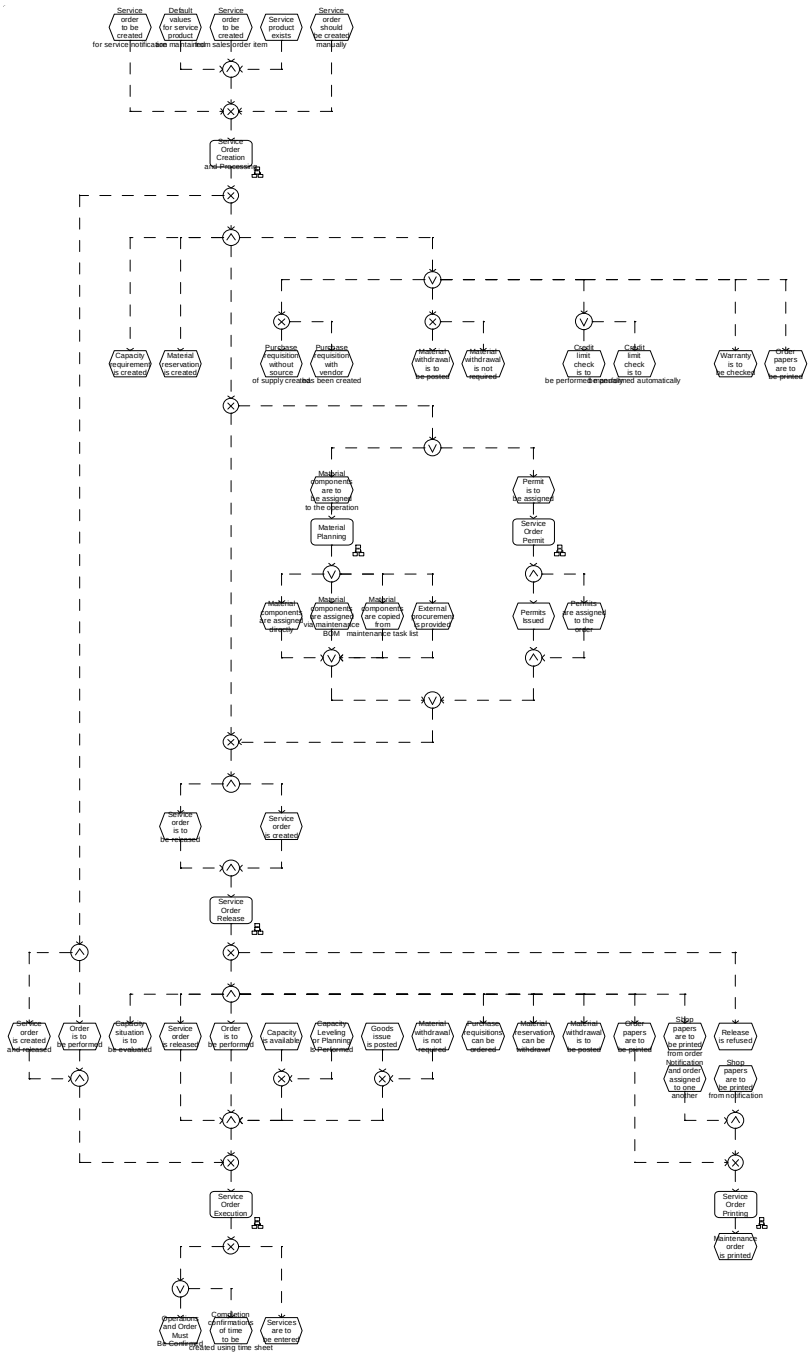


Figure B.14: Customer Service – Repairs Processing in Service Center (Inhouse) – Service Order (reduced size 11, unsound)

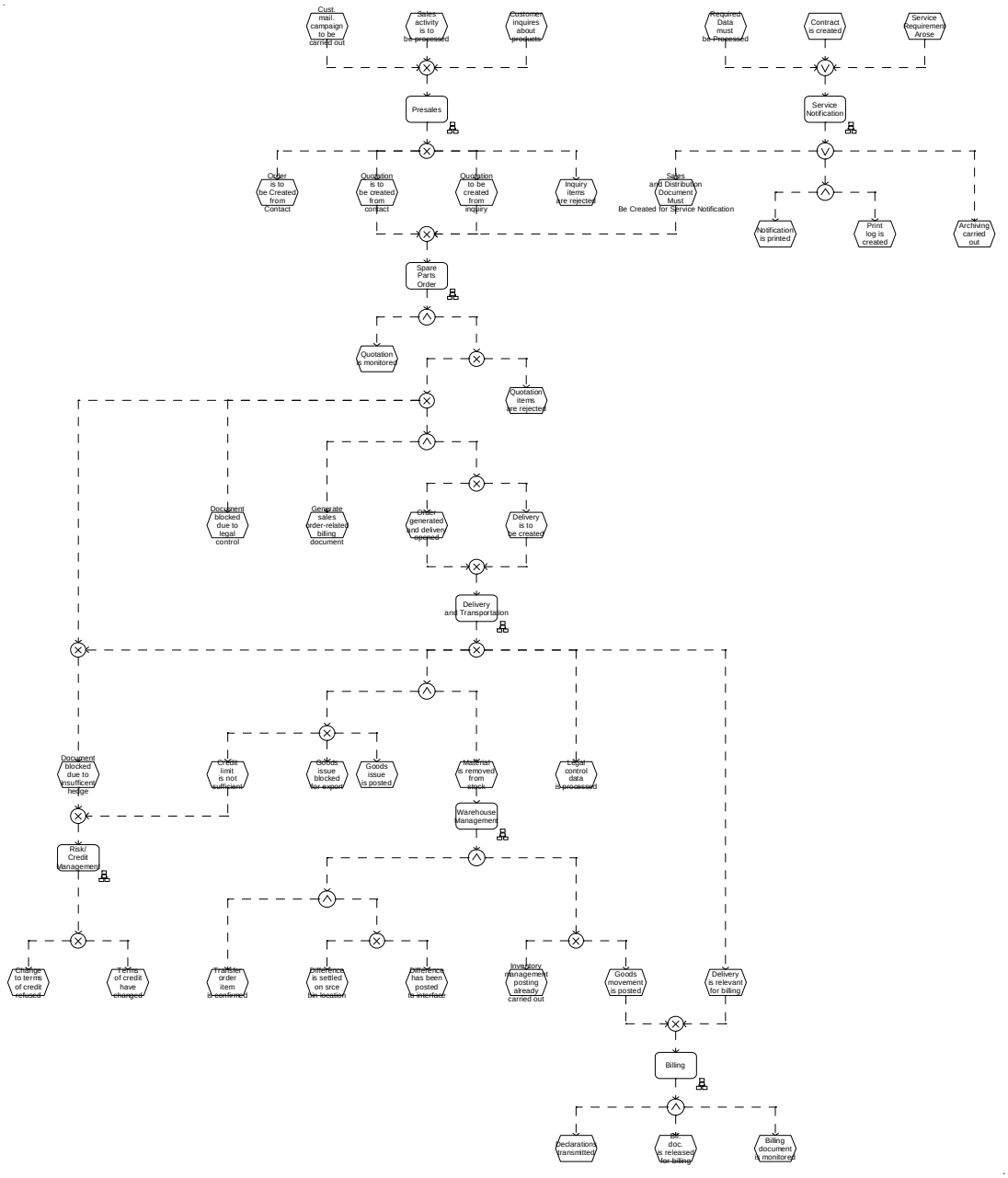


Figure B.15: Customer Service – Spare Parts Delivery Processing (reduced size 18, sound)

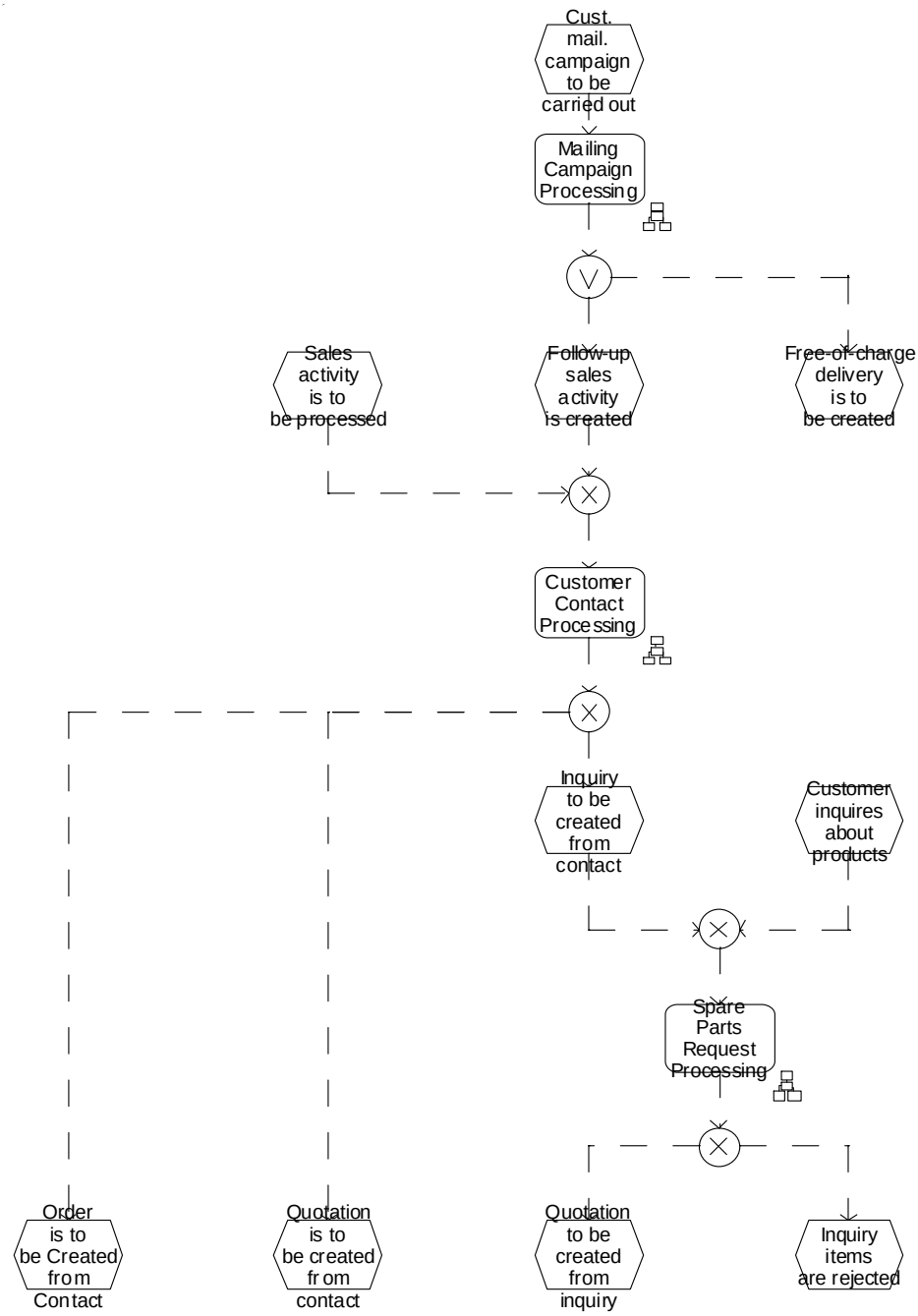


Figure B.16: Customer Service – Spare Parts Delivery Processing – Presales (reduced size 10, sound)

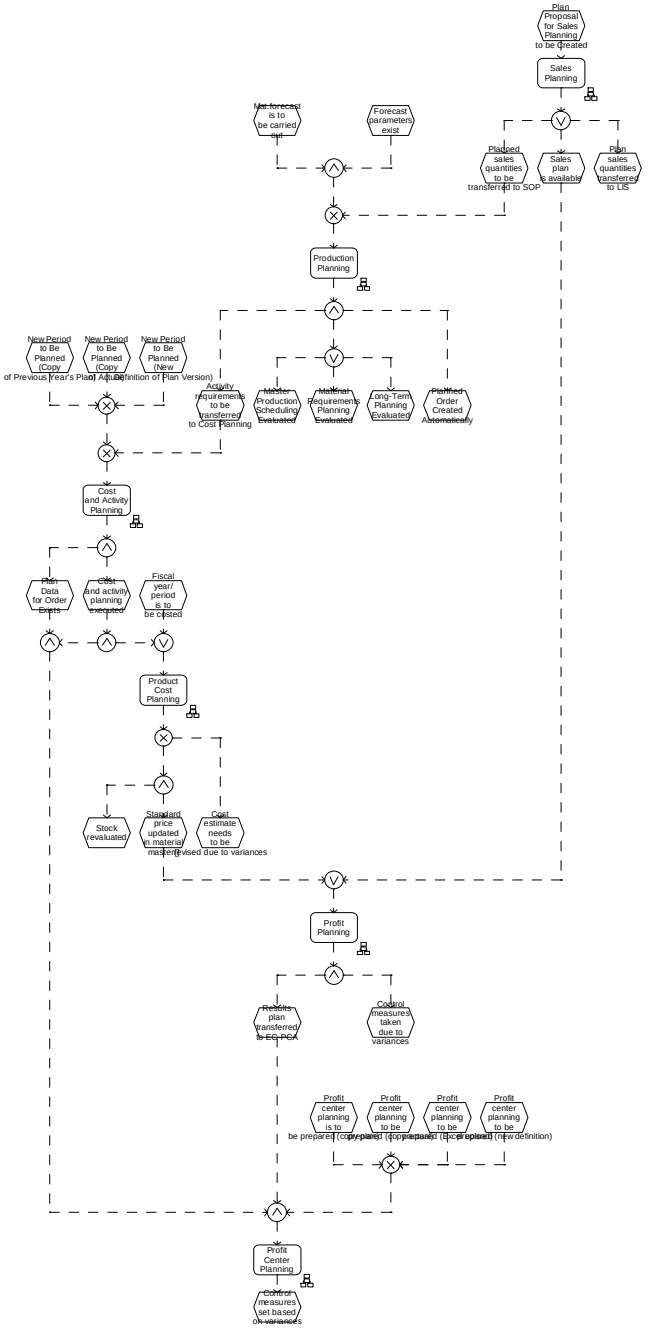


Figure B.17: Enterprise Controlling – Operational business planning (reduced size 14, unsound)

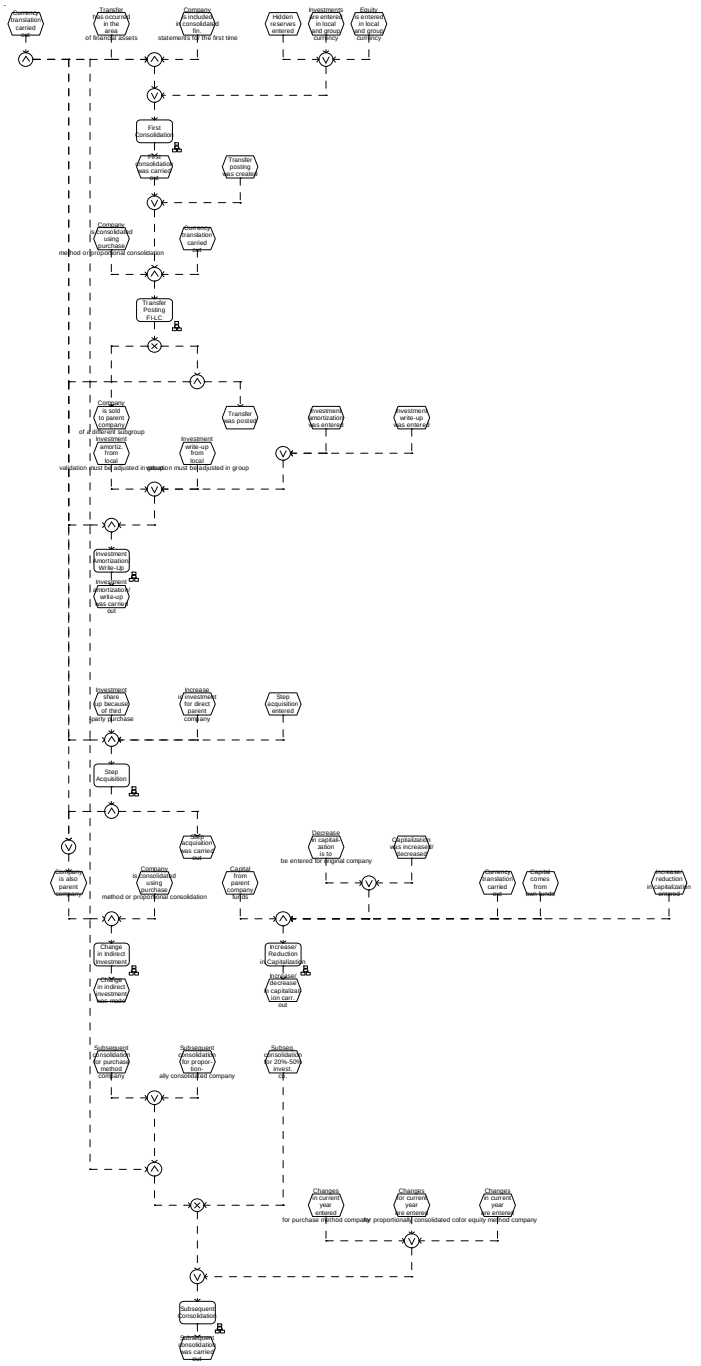


Figure B.18: Financial Accounting – Consolidation – Consolidation of Investments (reduced size 26, sound)

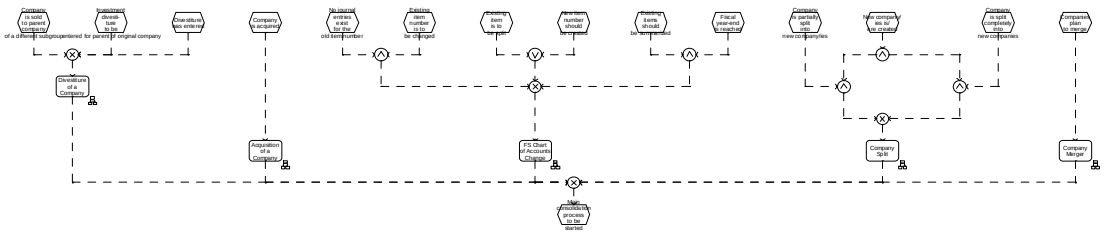


Figure B.19: Financial Accounting – Consolidation – Master Data Maintenance (reduced size 9, unsound)

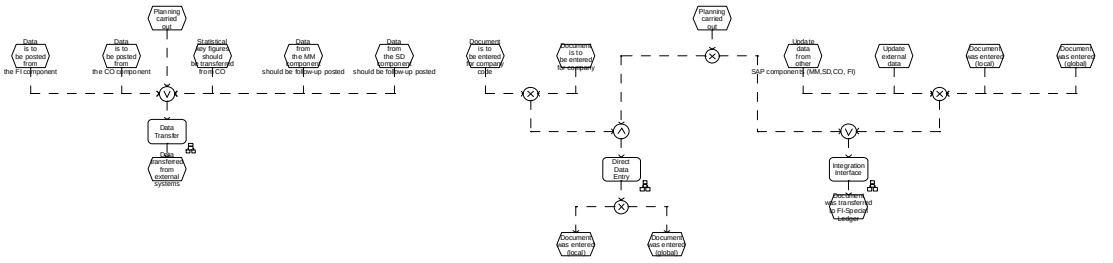


Figure B.20: Financial Accounting – Special Purpose Ledger – Actual Posting (reduced size 8, unsound)

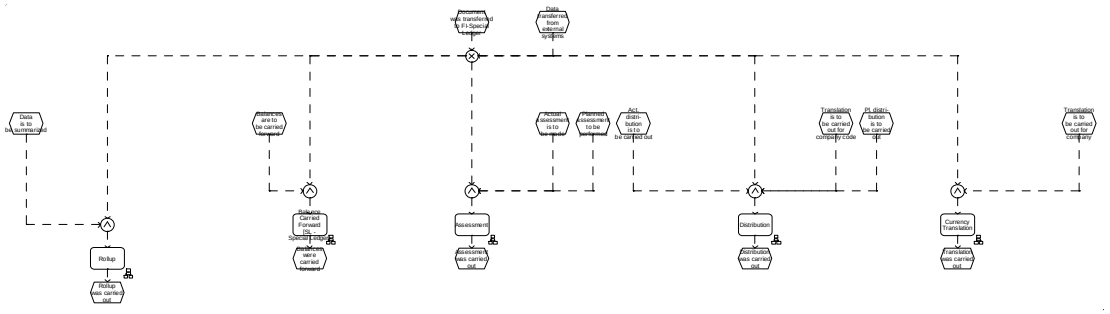


Figure B.21: Financial Accounting – Special Purpose Ledger – Periodic Processing (reduced size 17, unsound)

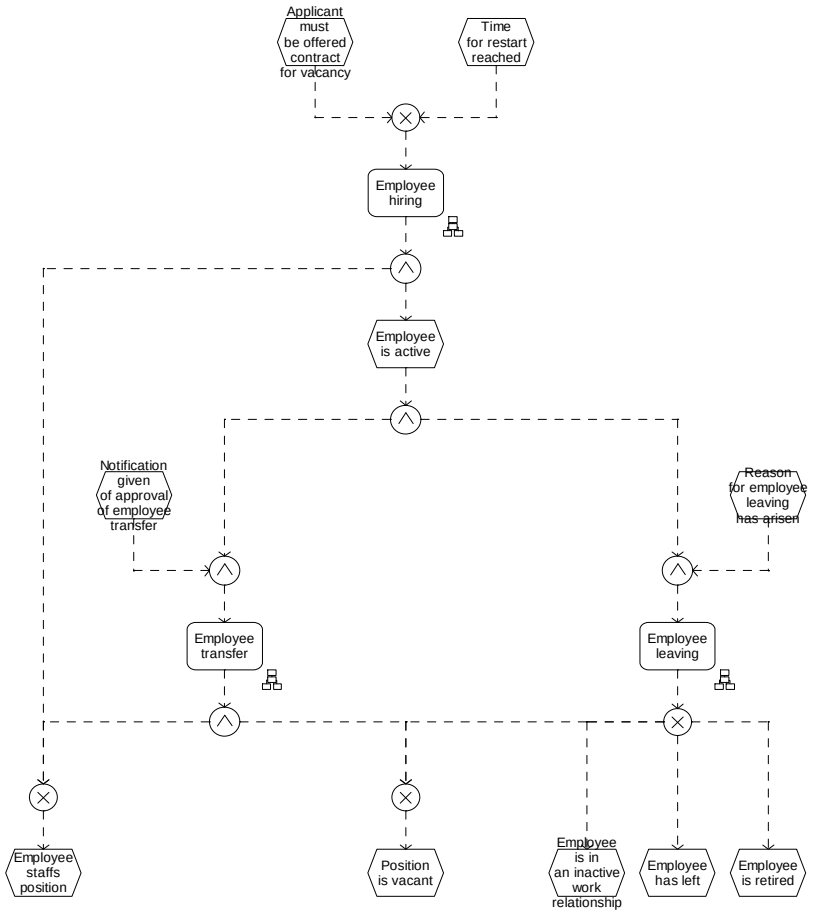


Figure B.22: Personnel Administration – Personnel Actions (reduced size 13, unsound)

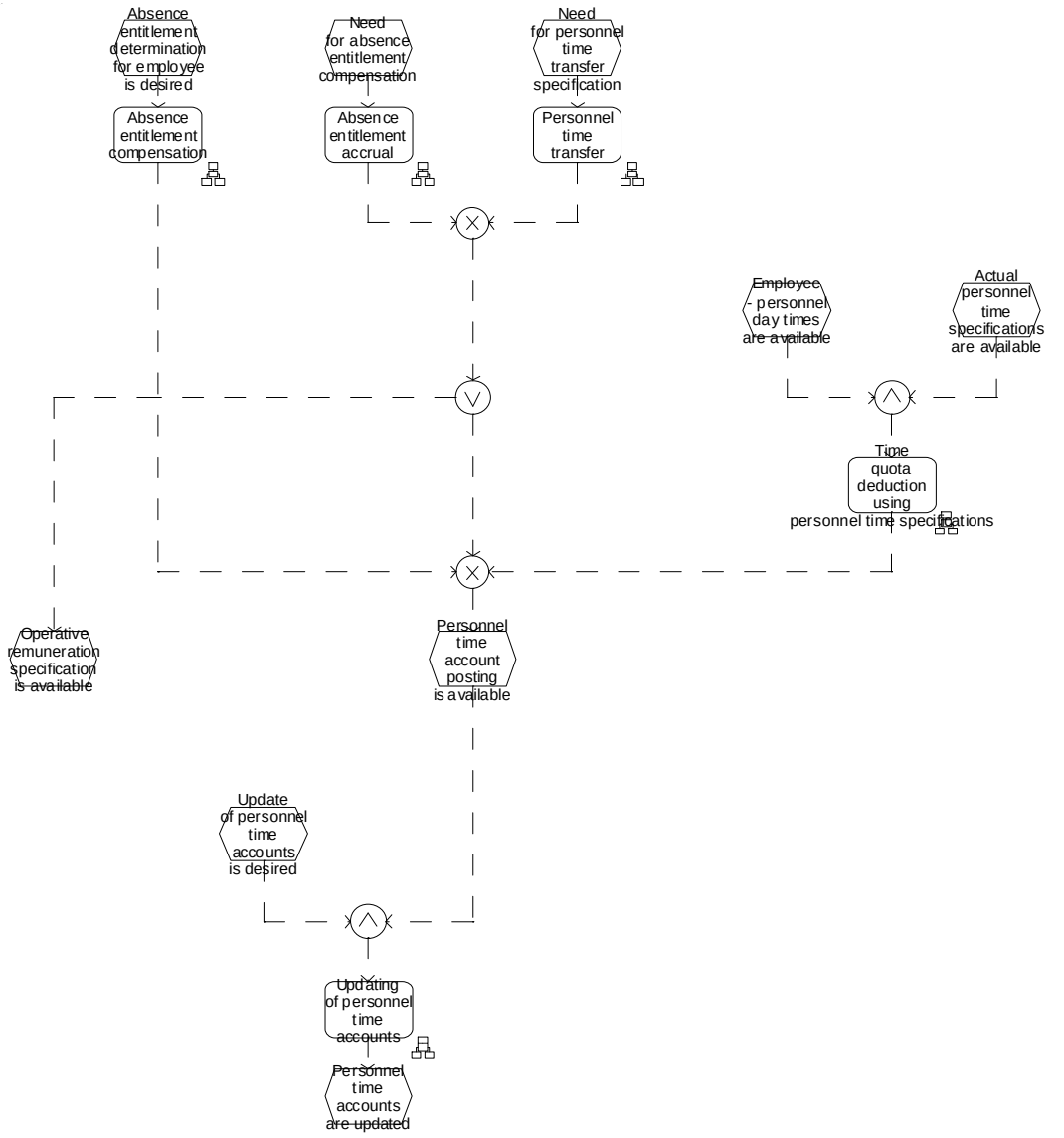


Figure B.23: Personnel Time Management – Personnel Time Management – Personnel time accounts administration (reduced size 8, unsound)

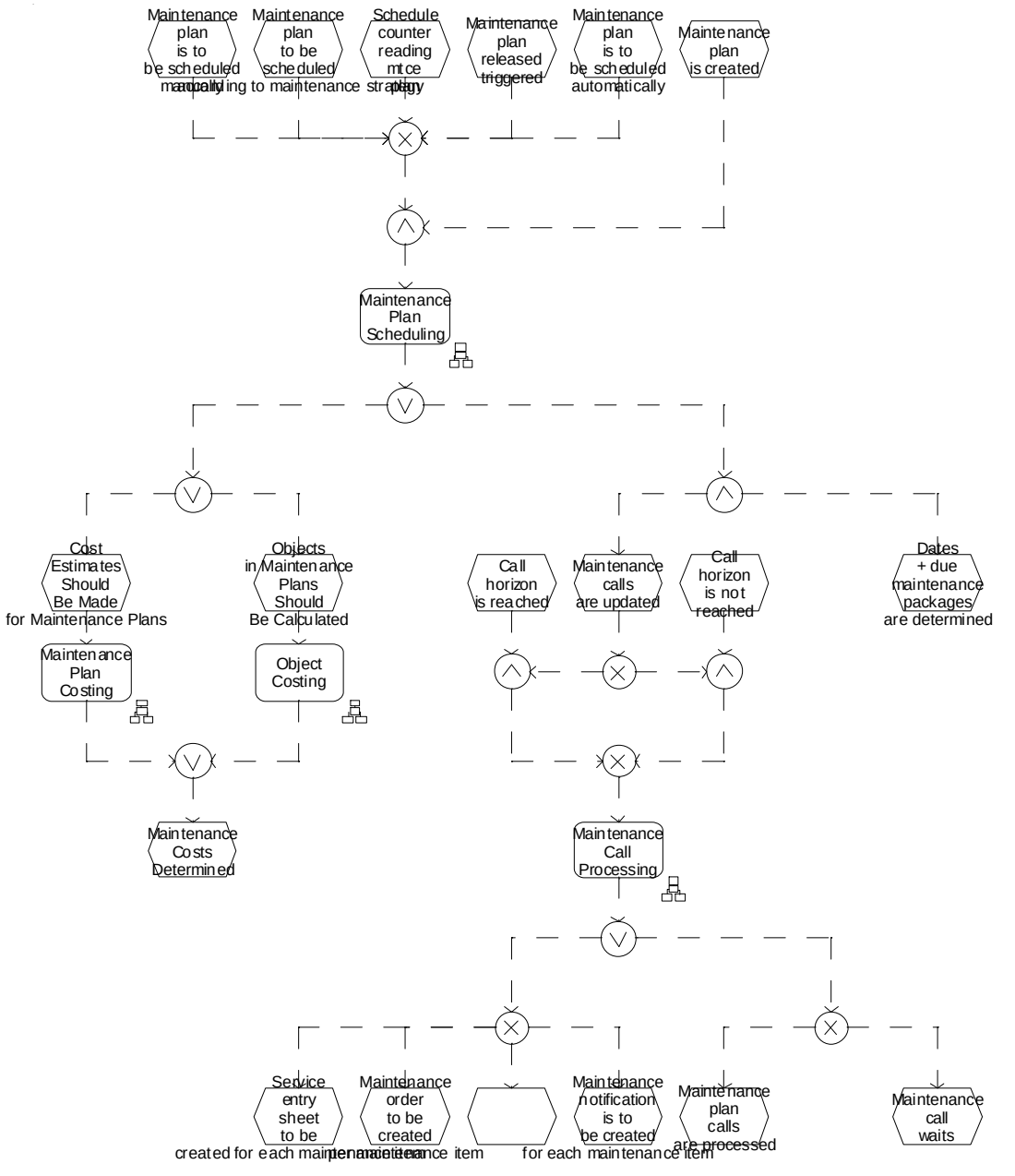


Figure B.24: Plant Maintenance – Planned Maintenance Processing – Maintenance Planning (reduced size 10, unsound)

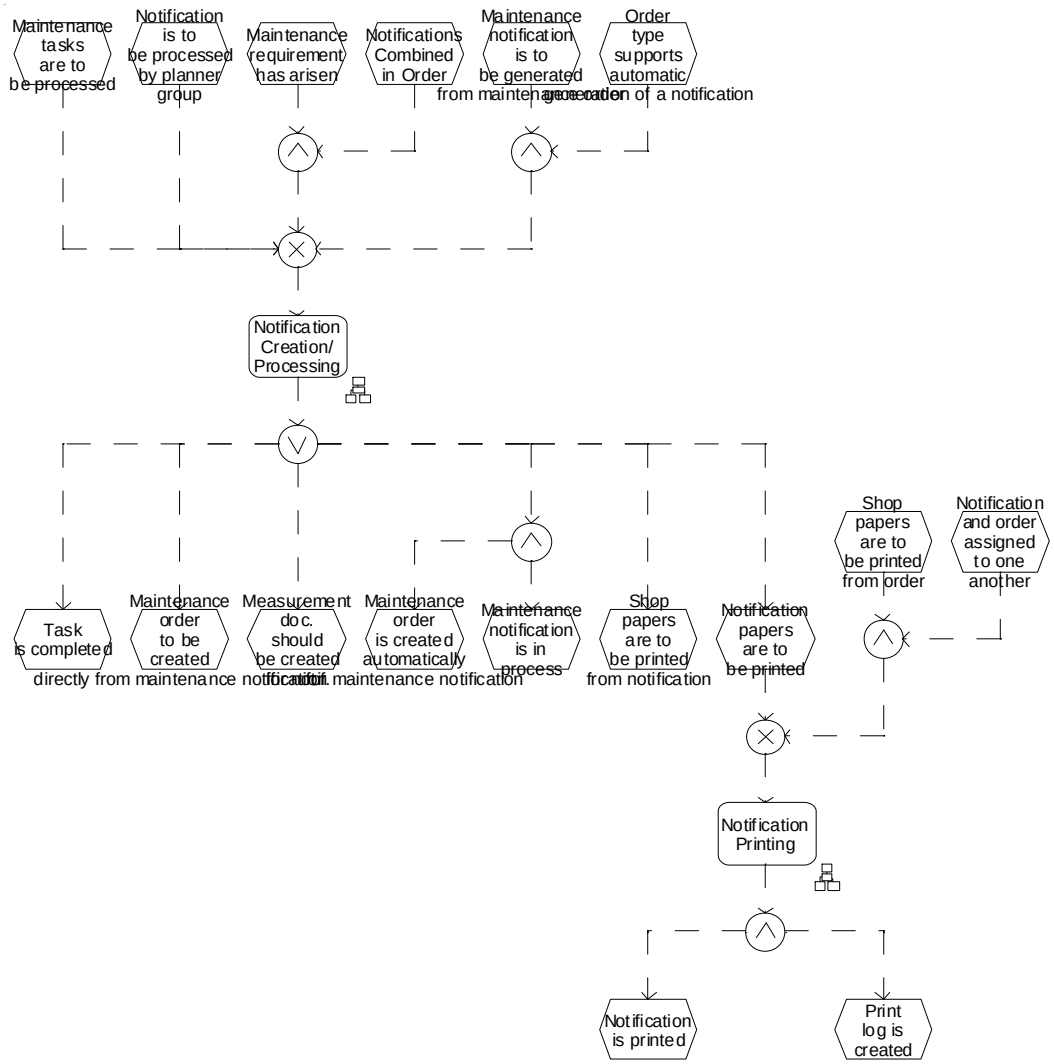


Figure B.25: Plant Maintenance – Planned Maintenance Processing – Notification (reduced size 6, sound)

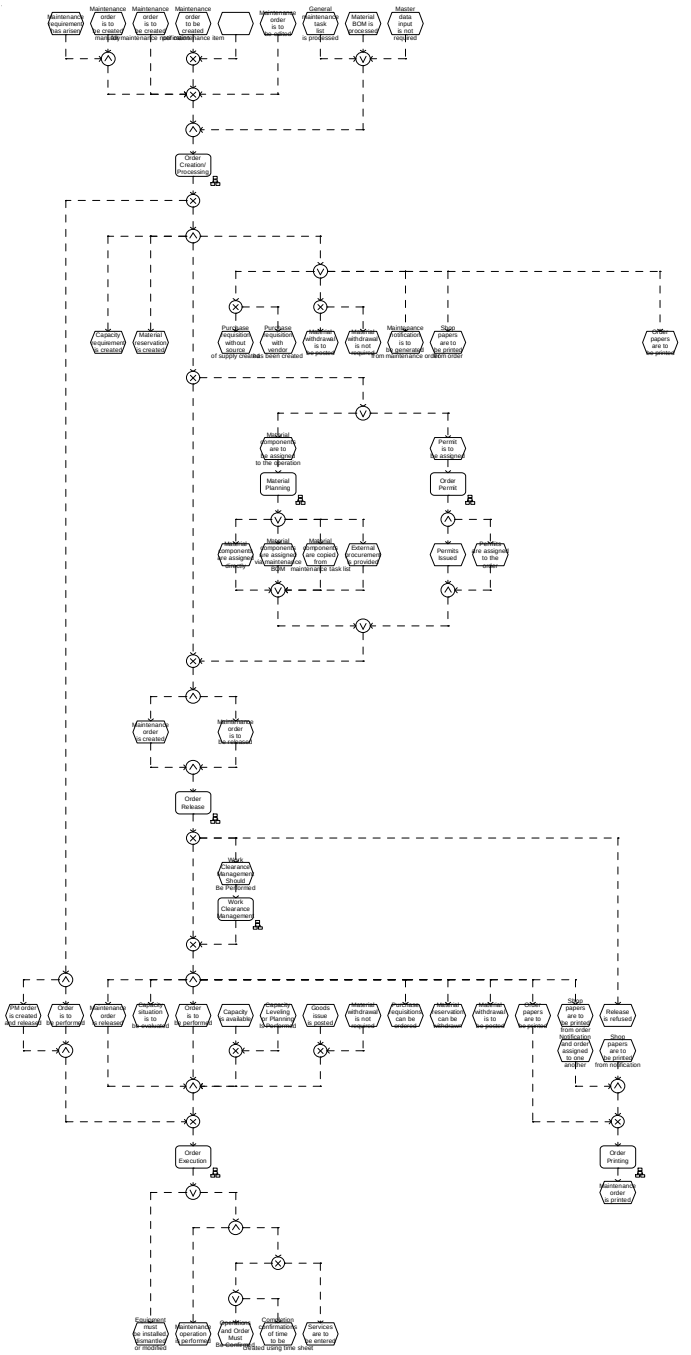


Figure B.26: Plant Maintenance – Planned Maintenance Processing – Order (reduced size 11, unsound)

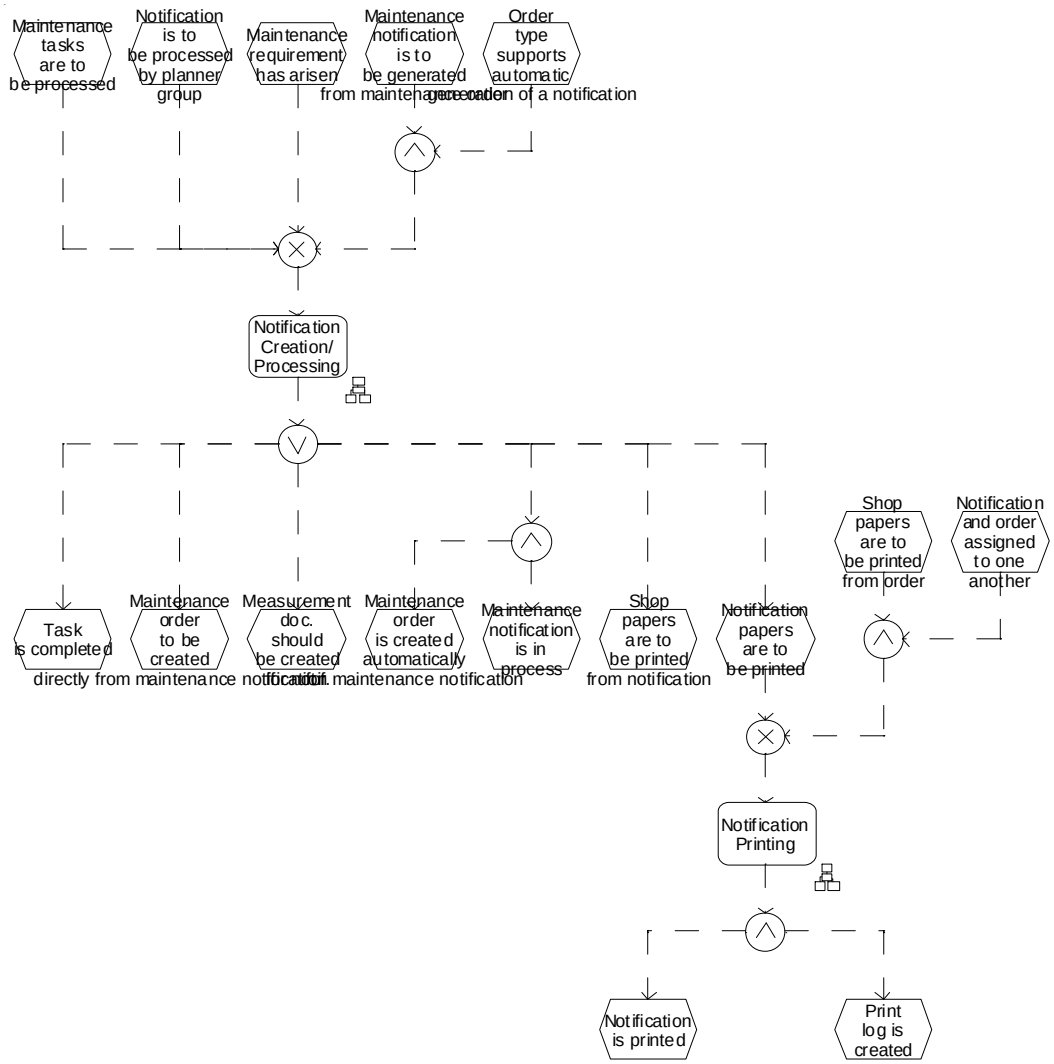


Figure B.27: Plant Maintenance – Project-Based Maintenance Processing – Notification (reduced size 6, sound)

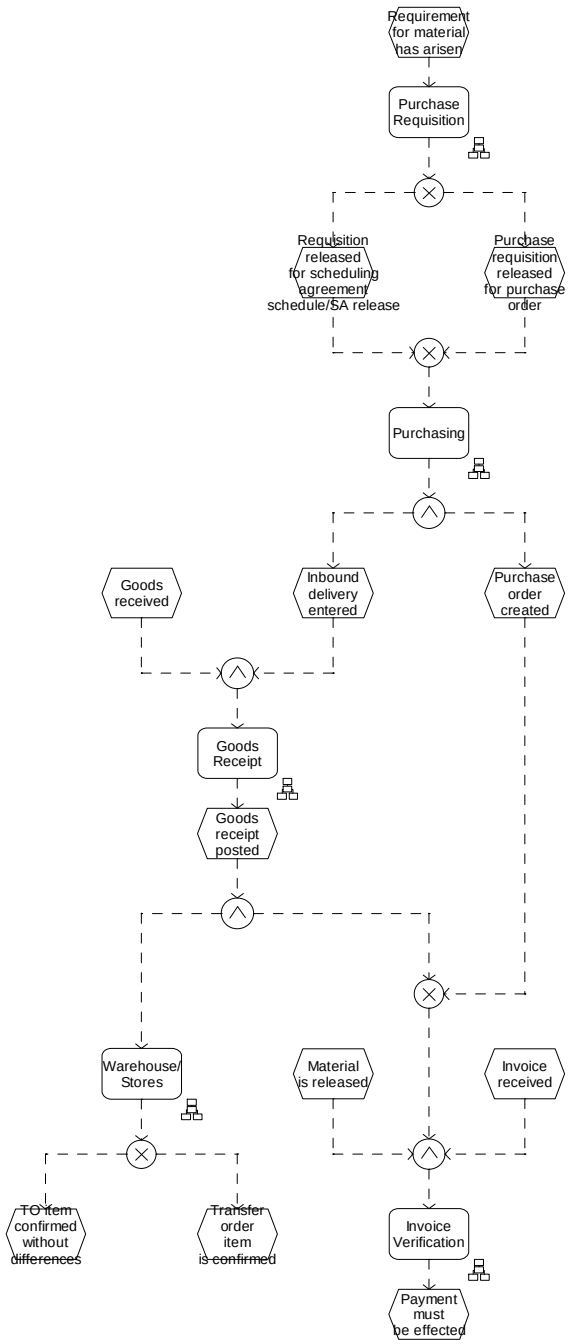


Figure B.28: Procurement – Internal Procurement (reduced size 8, unsound)

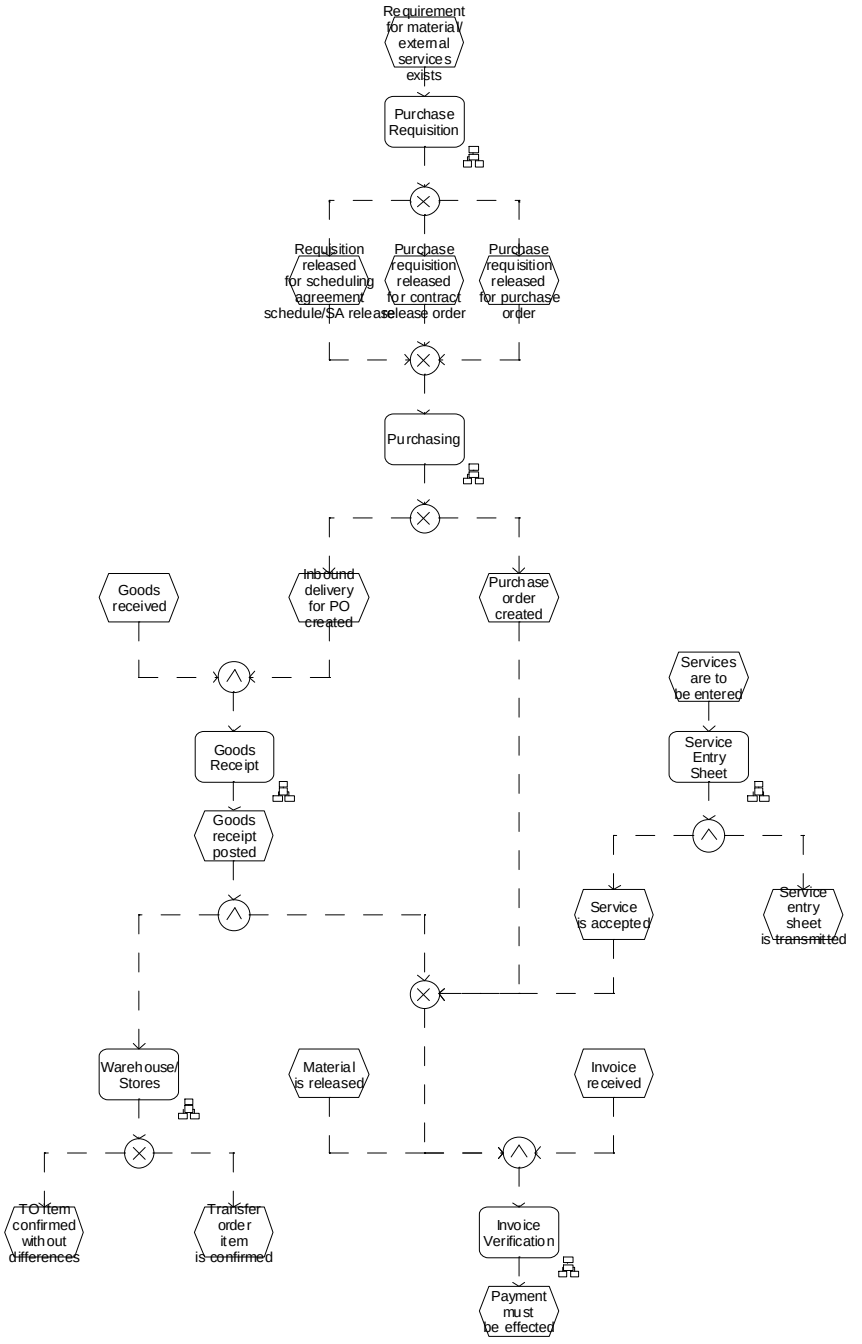


Figure B.29: Procurement – Procurement of Materials and External Services (reduced size 9, unsound)

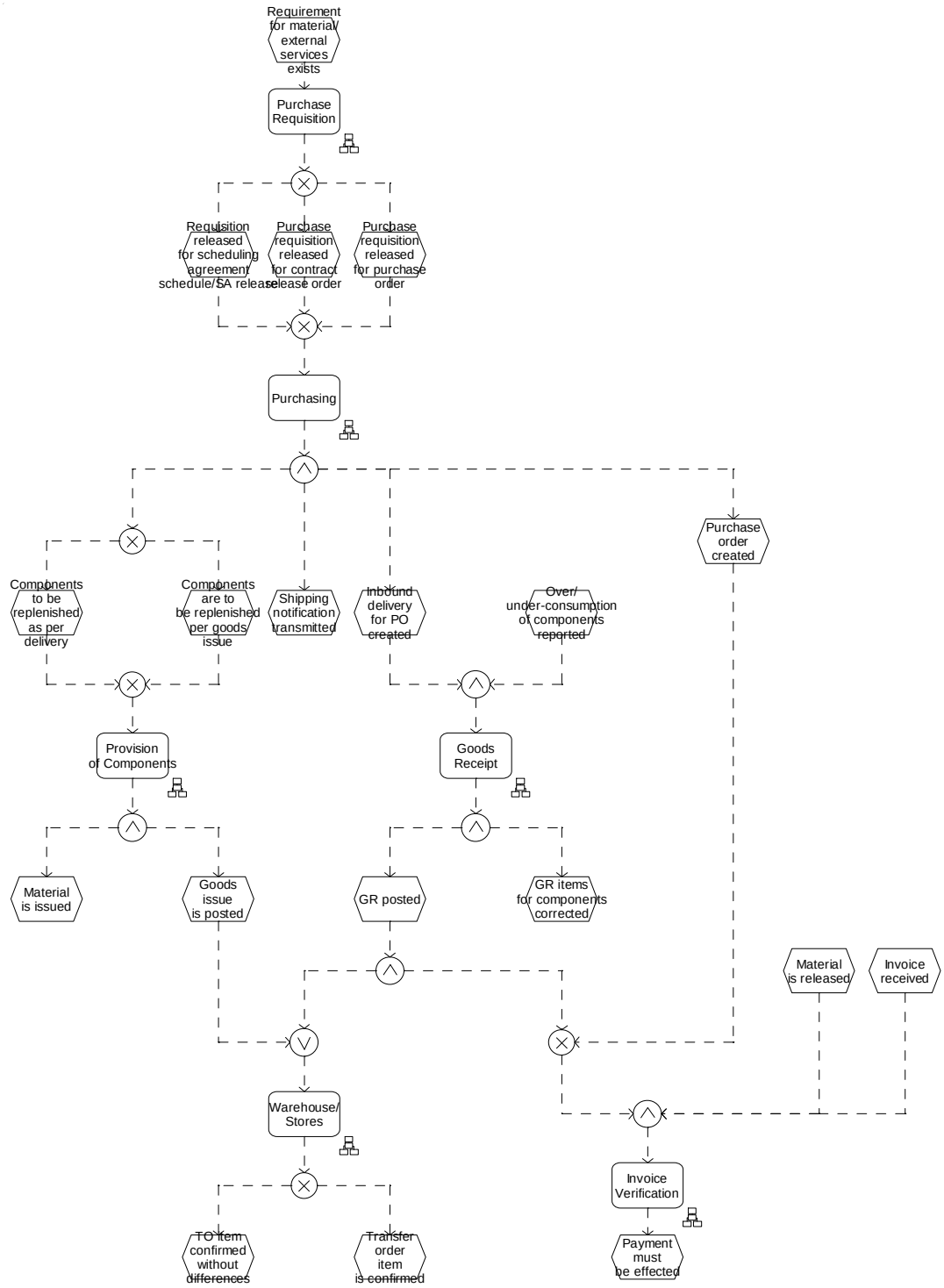


Figure B.30: Procurement – Procurement via Subcontracting (reduced size 11, unsound)

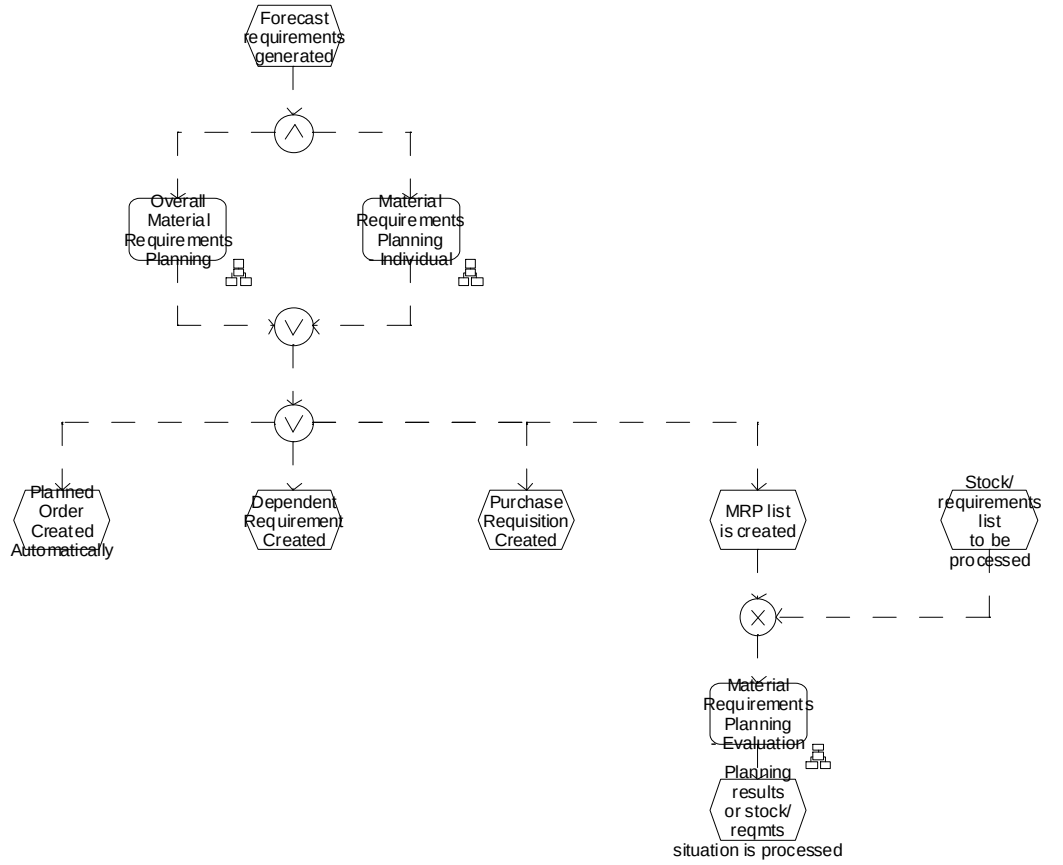


Figure B.31: Production Planning and Procurement Planning – Consumption-Driven Planning – Material Requirements Planning (reduced size 6, sound)

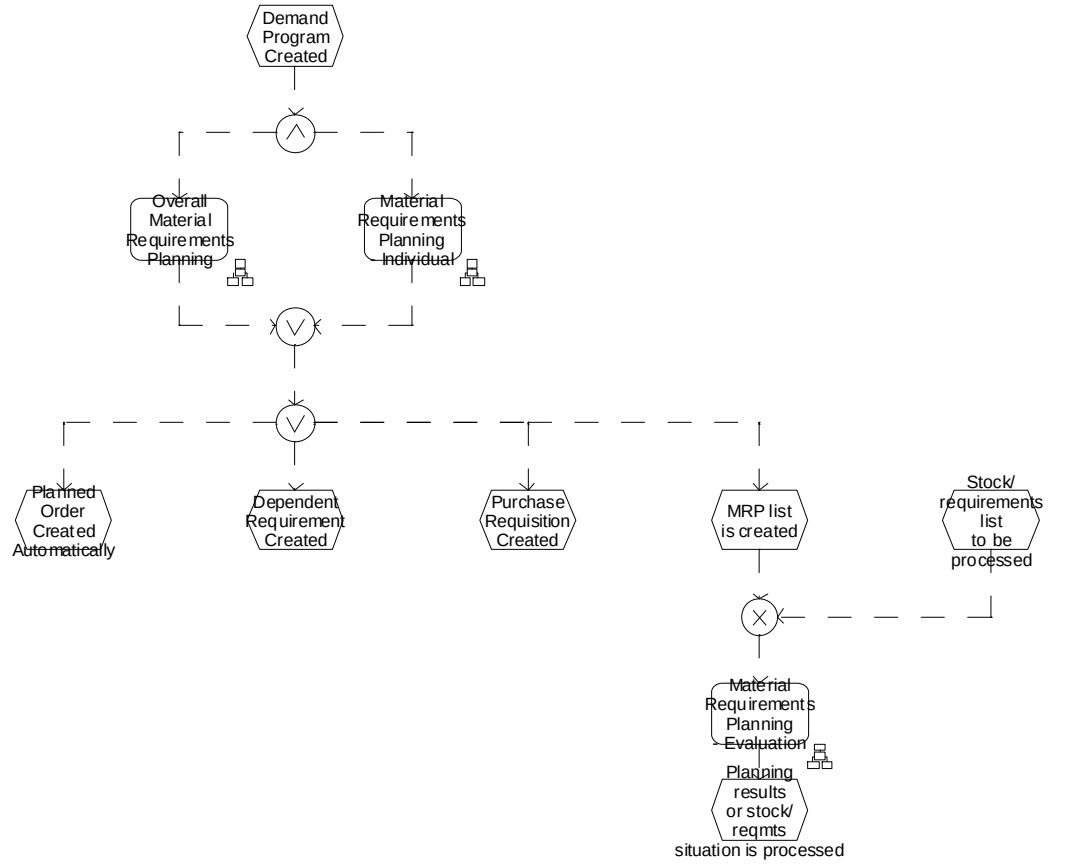


Figure B.32: Production Planning and Procurement Planning – Market-Oriented Planning (reduced size 11, sound)

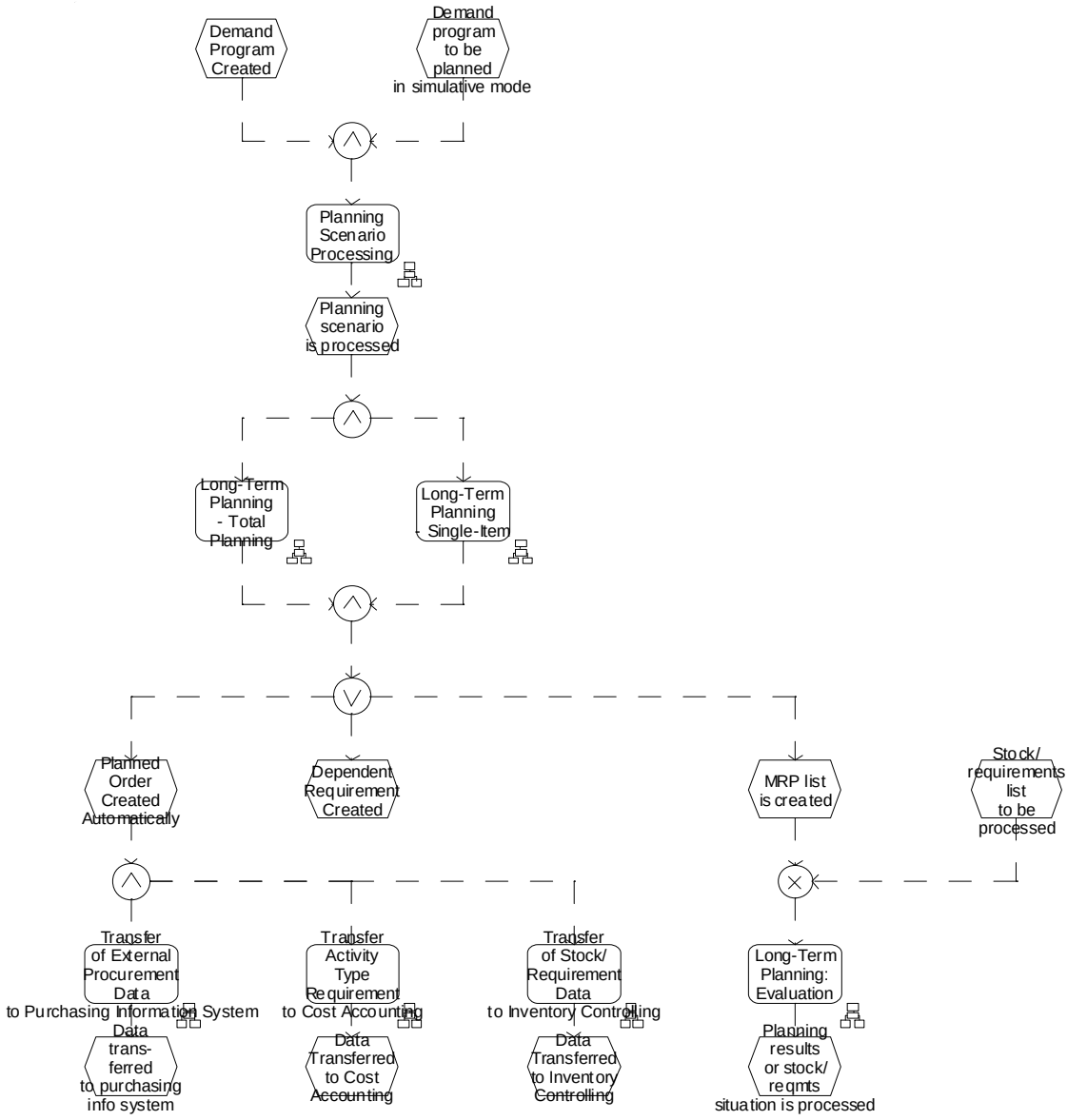


Figure B.33: Production Planning and Procurement Planning – Market-Oriented Planning – Long-Term Planning (reduced size 6, sound)

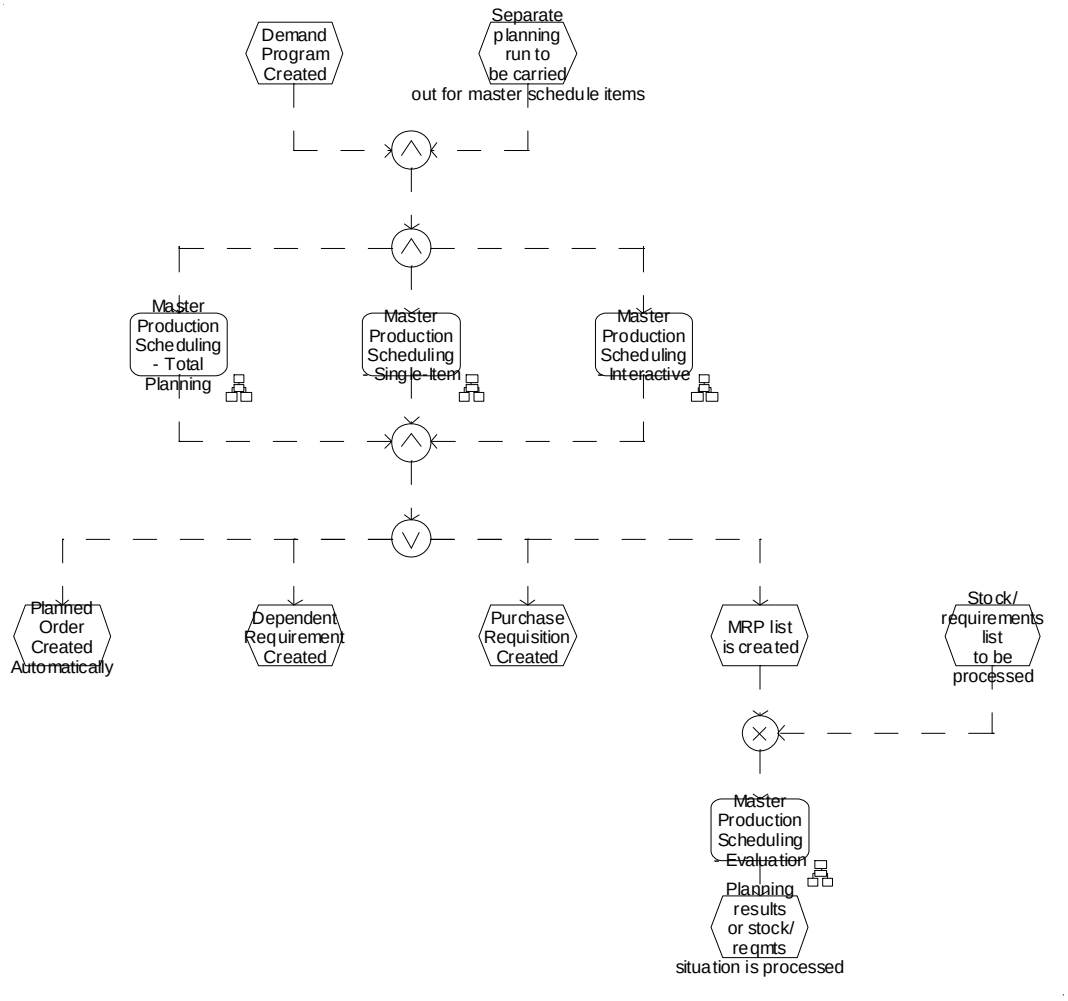


Figure B.34: Production Planning and Procurement Planning – Market-Oriented Planning – Master Production Scheduling (reduced size 6, sound)

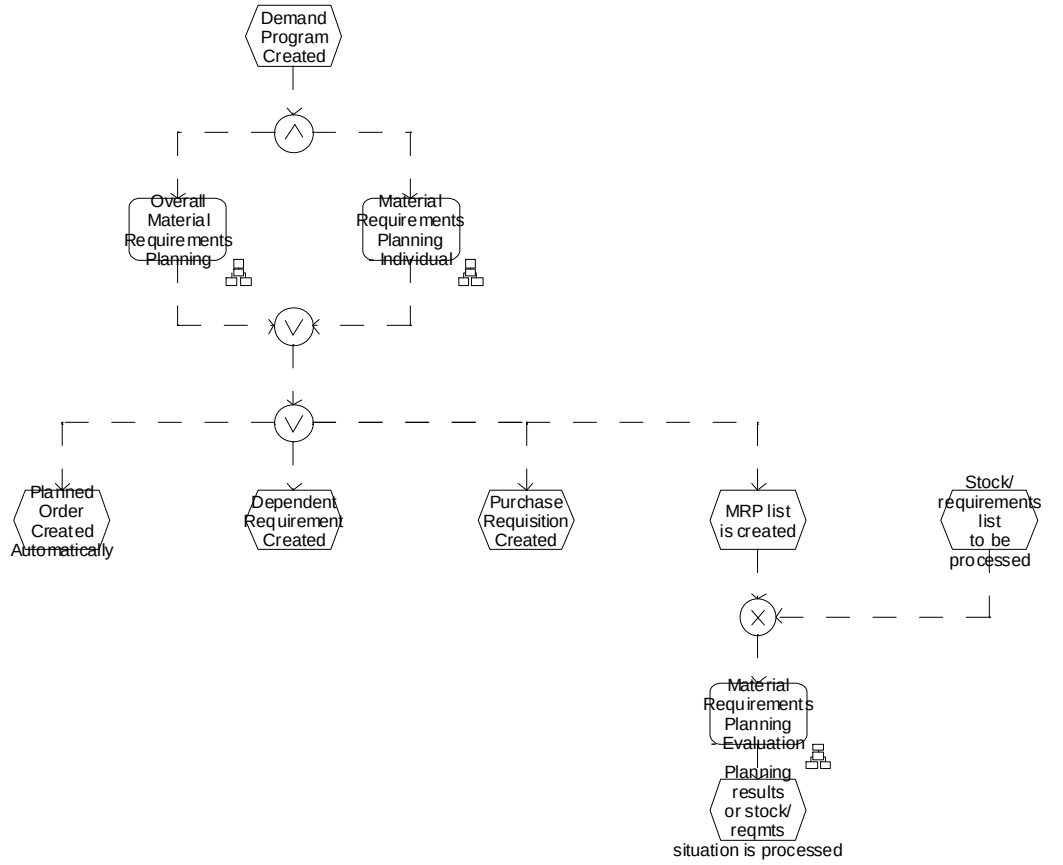


Figure B.35: Production Planning and Procurement Planning – Market-Oriented Planning – Material Requirements Planning (reduced size 6, sound)

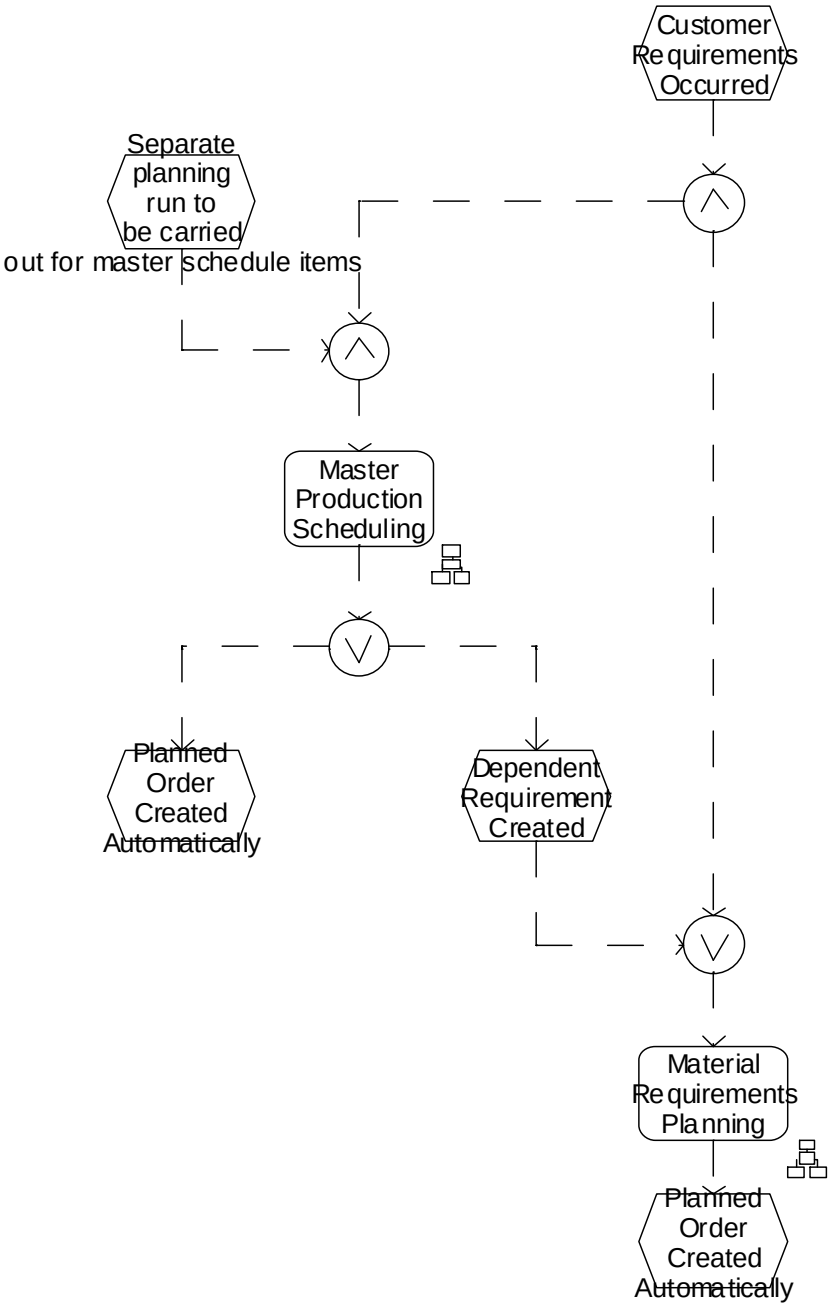


Figure B.36: Production Planning and Procurement Planning – Sales Order Oriented Planning (reduced size 8, sound)

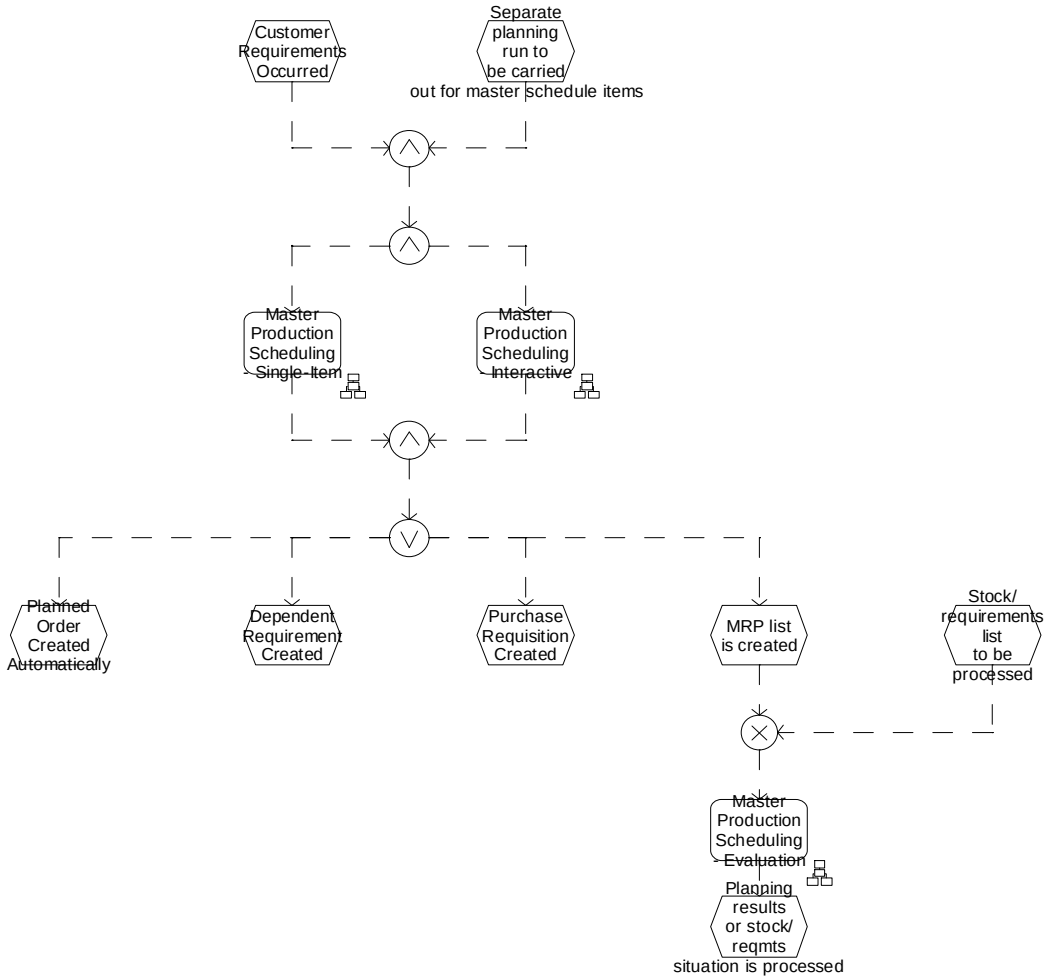


Figure B.37: Production Planning and Procurement Planning – Sales Order Oriented Planning – Master Production Scheduling (reduced size 6, sound)

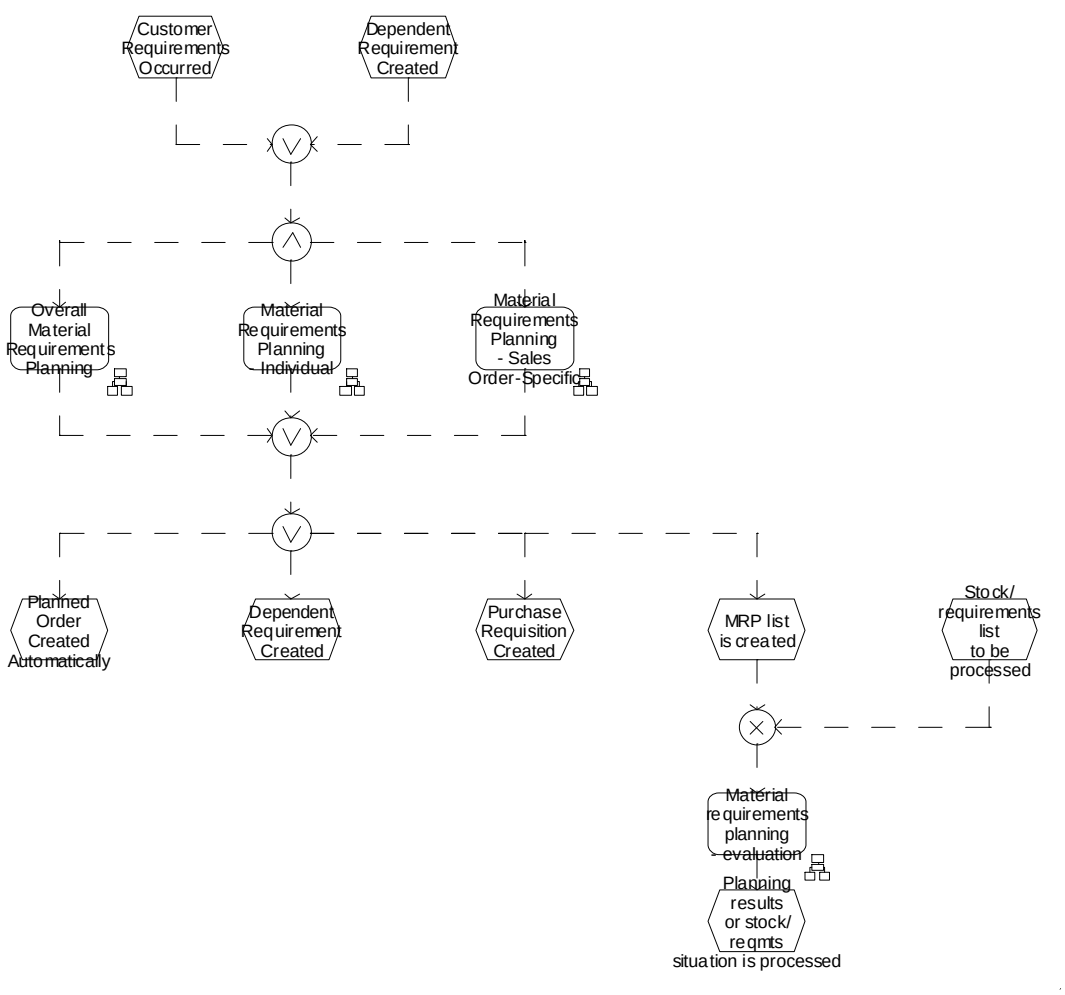


Figure B.38: Production Planning and Procurement Planning – Sales Order Oriented Planning – Material Requirements Planning (reduced size 6, sound)

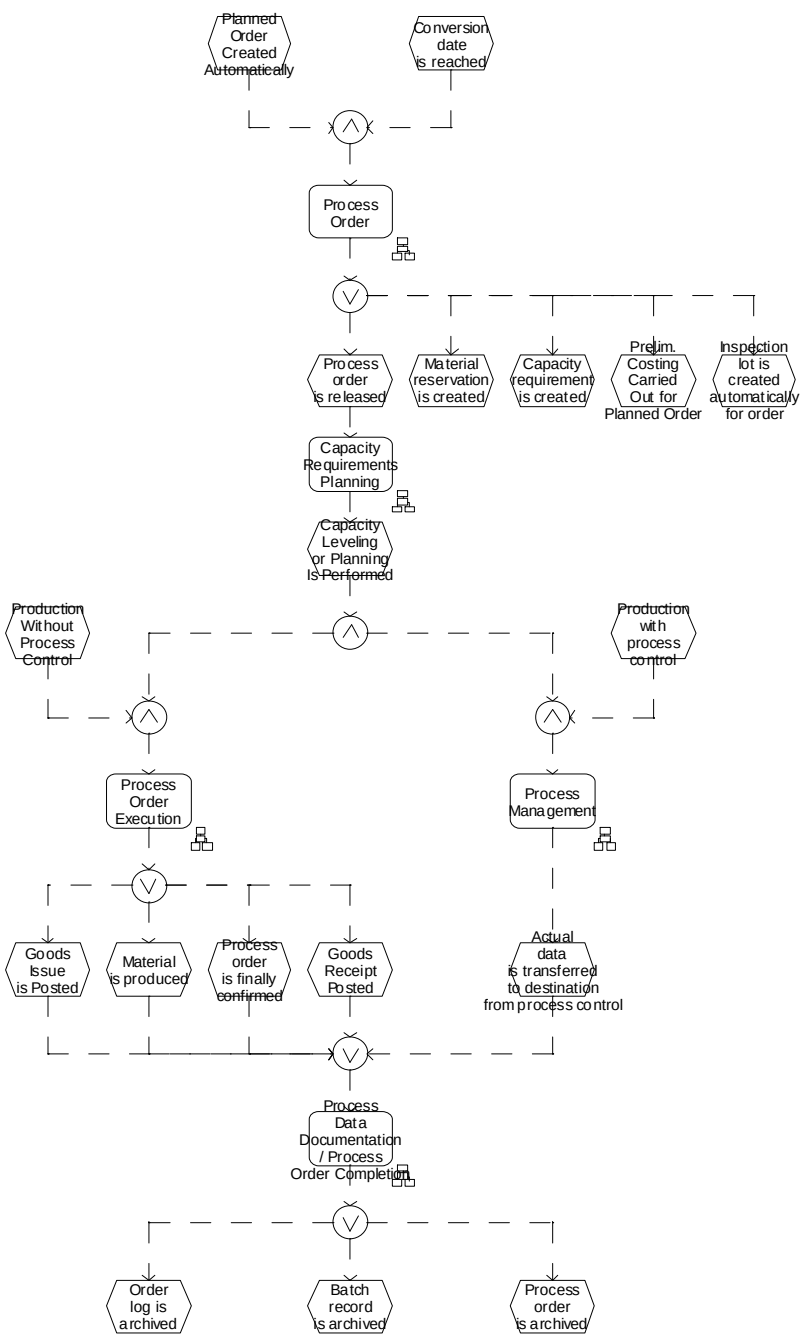


Figure B.39: Production – Process Manufacturing (reduced size 10, unsound)

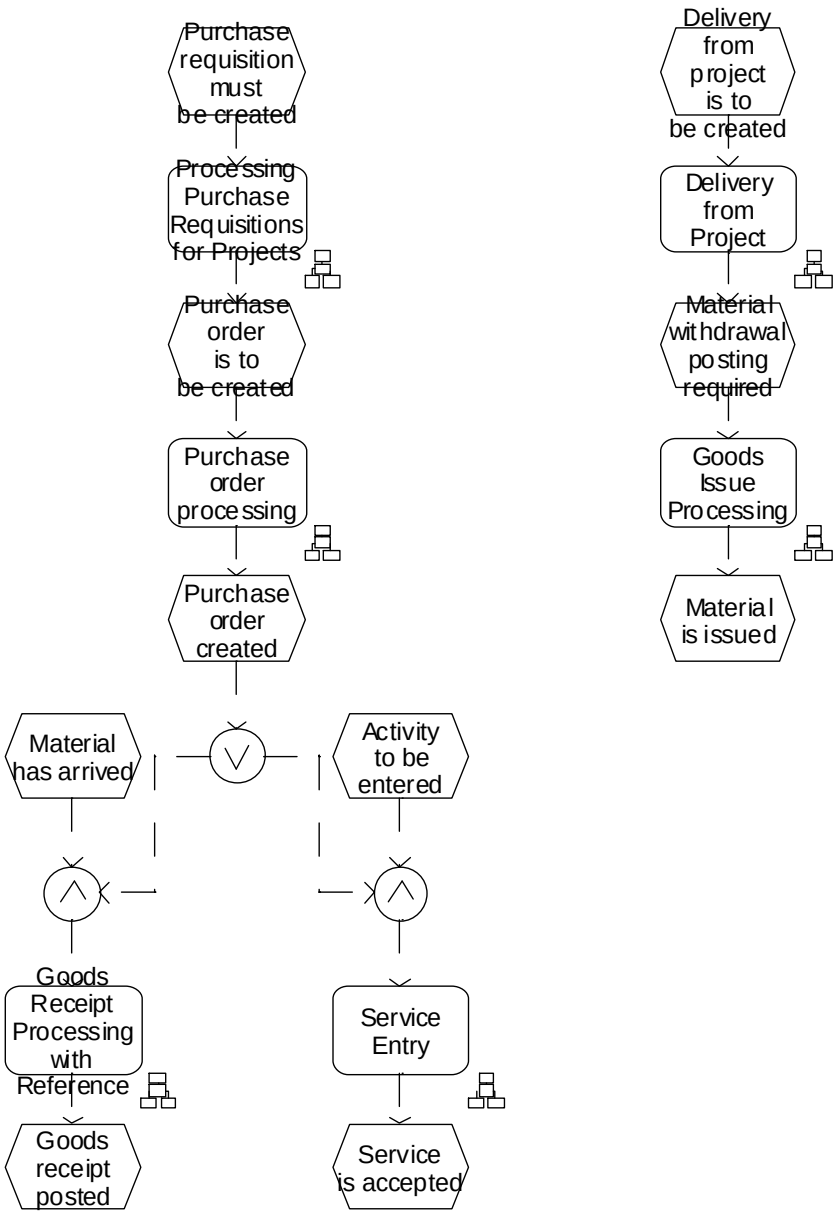


Figure B.40: Project Management – Execution – Materials Procurement and Service Processing (reduced size 8, unsound)

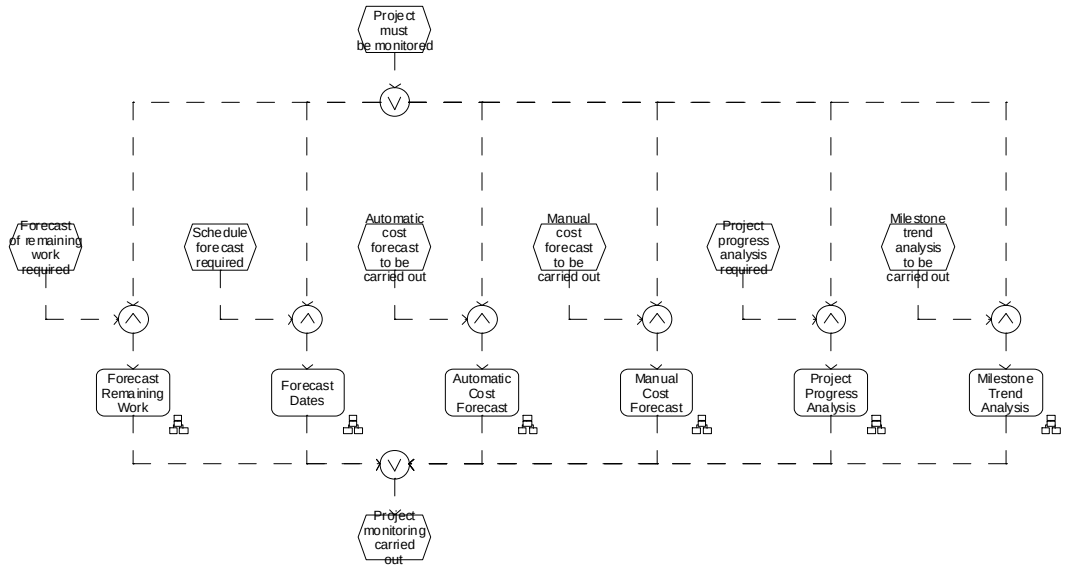


Figure B.41: Project Management – Execution – Project Monitoring and Controlling (reduced size 16, unsound)

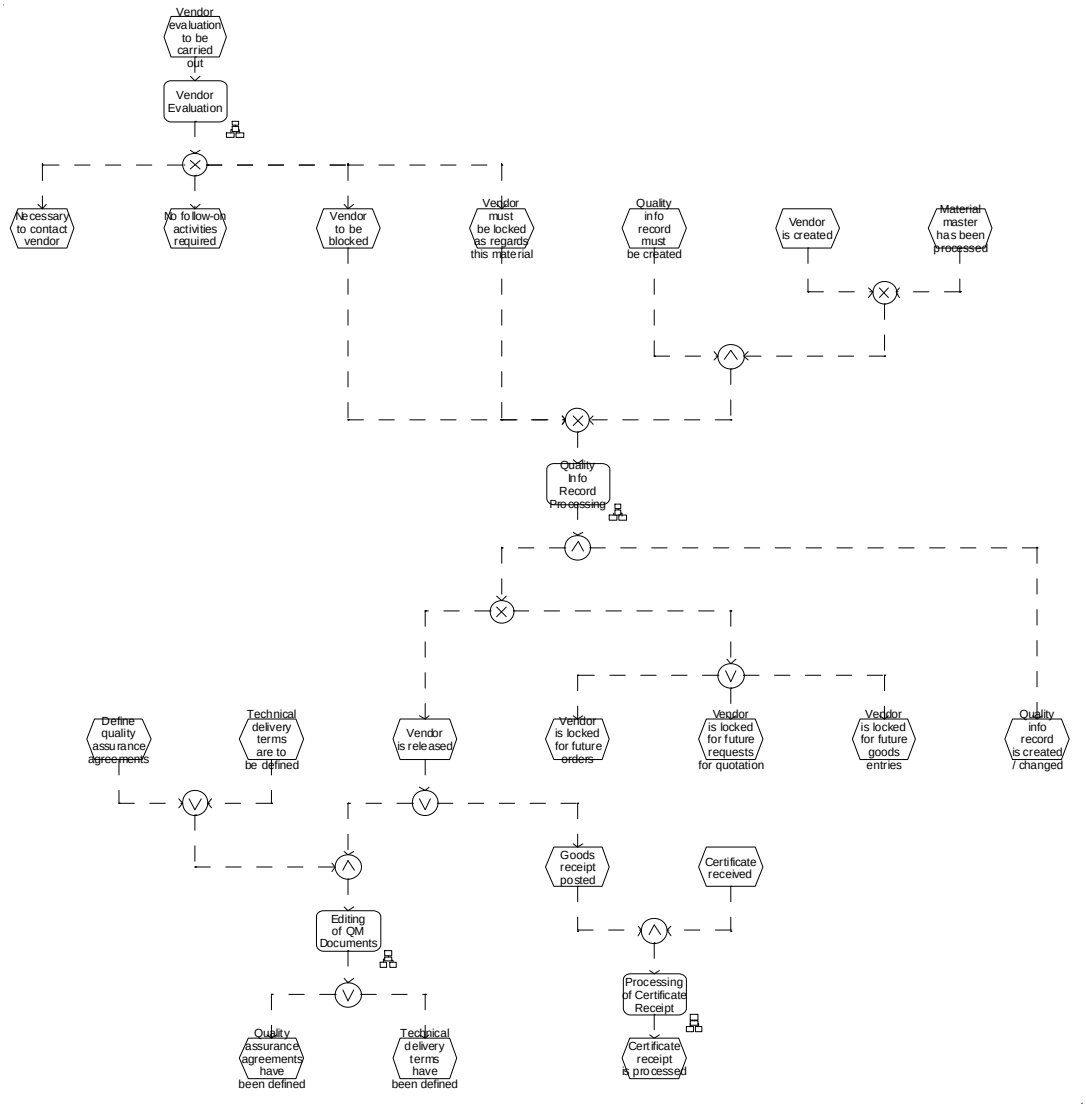


Figure B.42: Quality Management – QM in Materials Management – Procurement and Purchasing (reduced size 14, unsound)

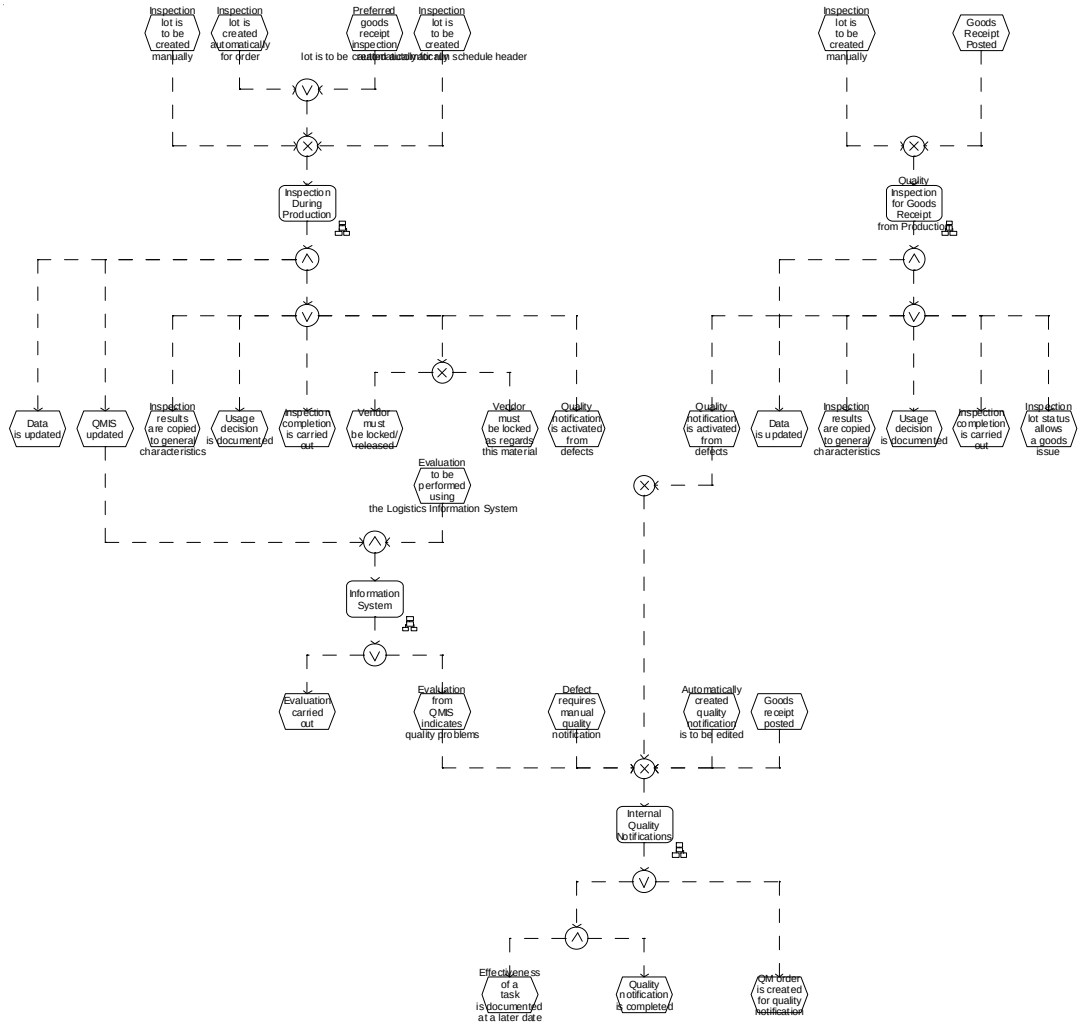


Figure B.43: Quality Management – QM in Production (reduced size 9, sound)

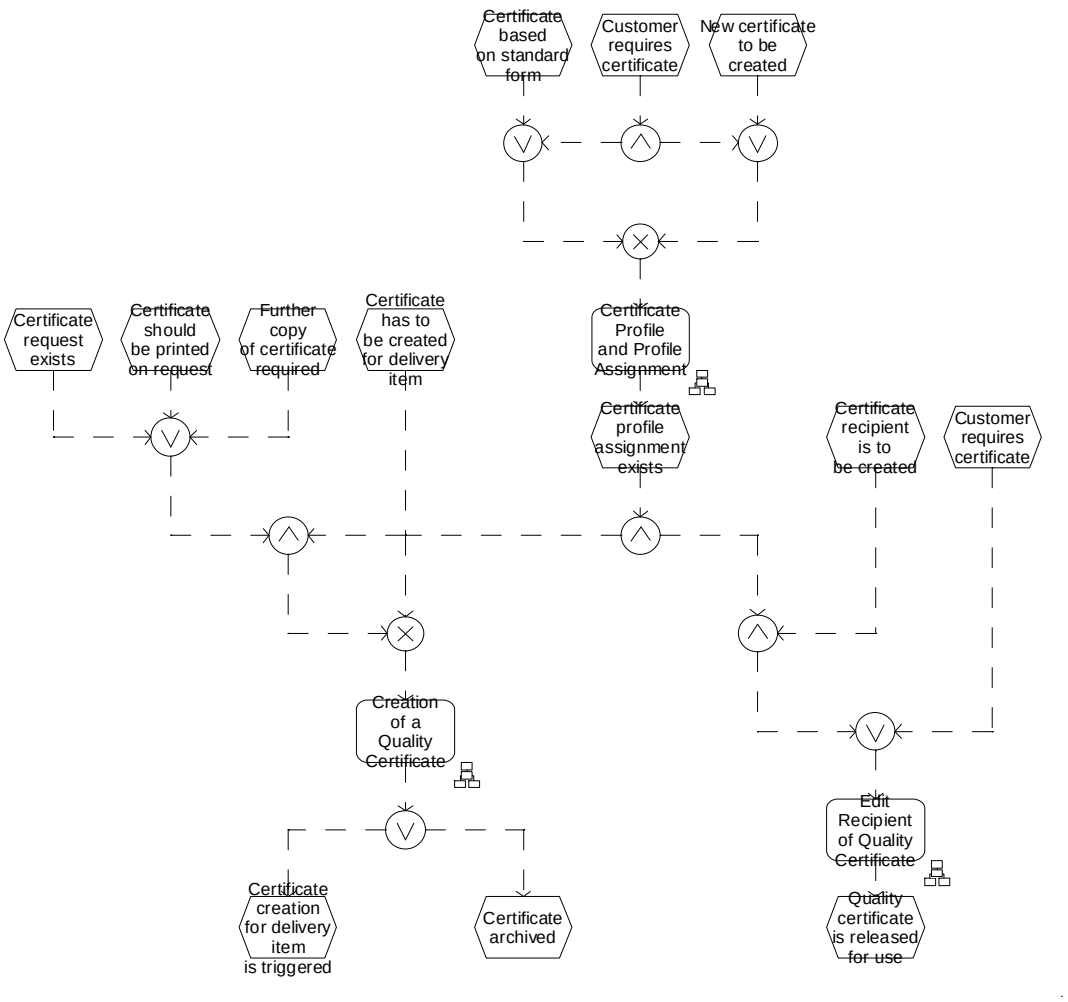


Figure B.44: Quality Management – QM in Sales and Distribution – Certificate Creation (reduced size 16, unsound)

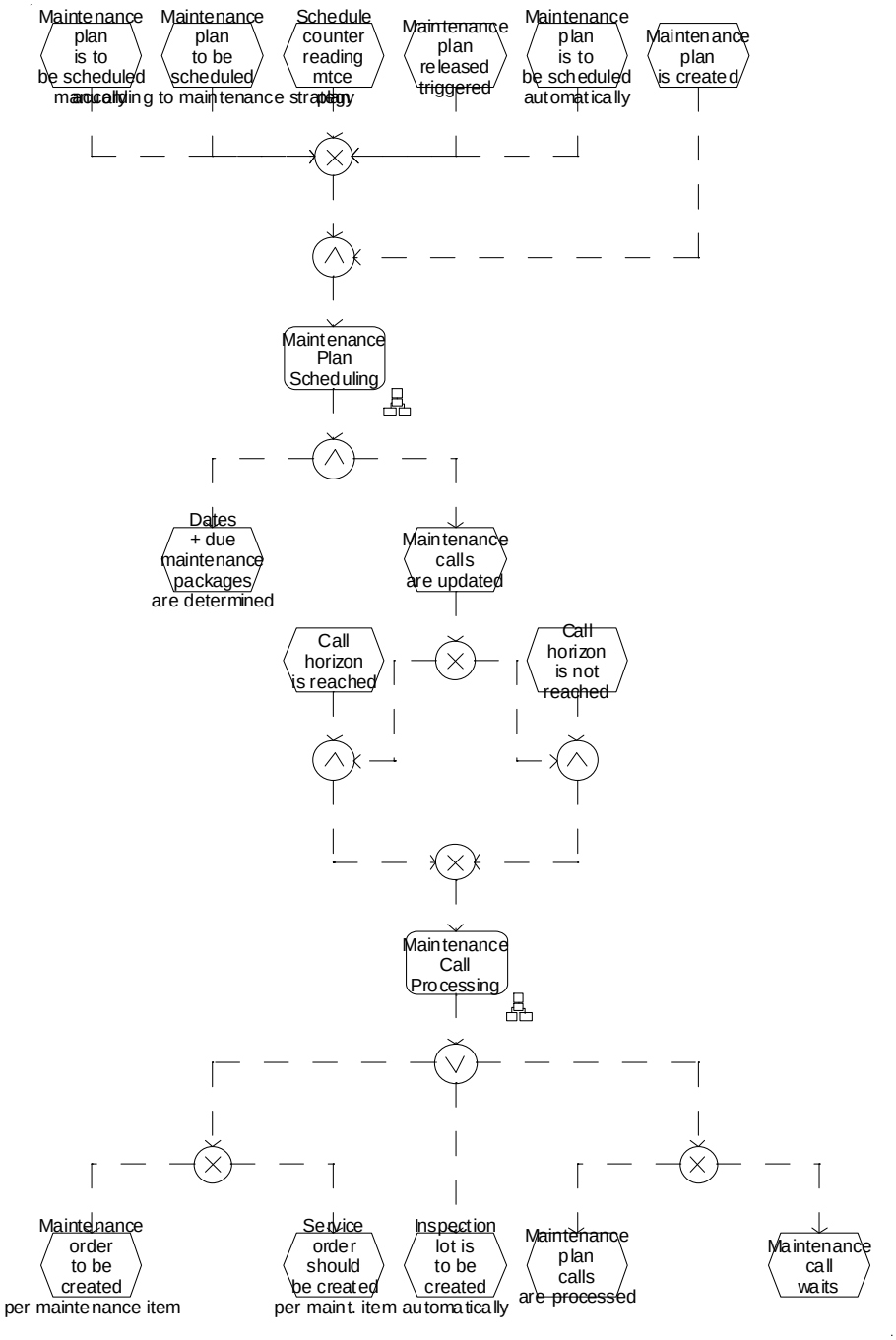


Figure B.45: Quality Management – Test Equipment Management – Maintenance Planning (reduced size 8, unsound)

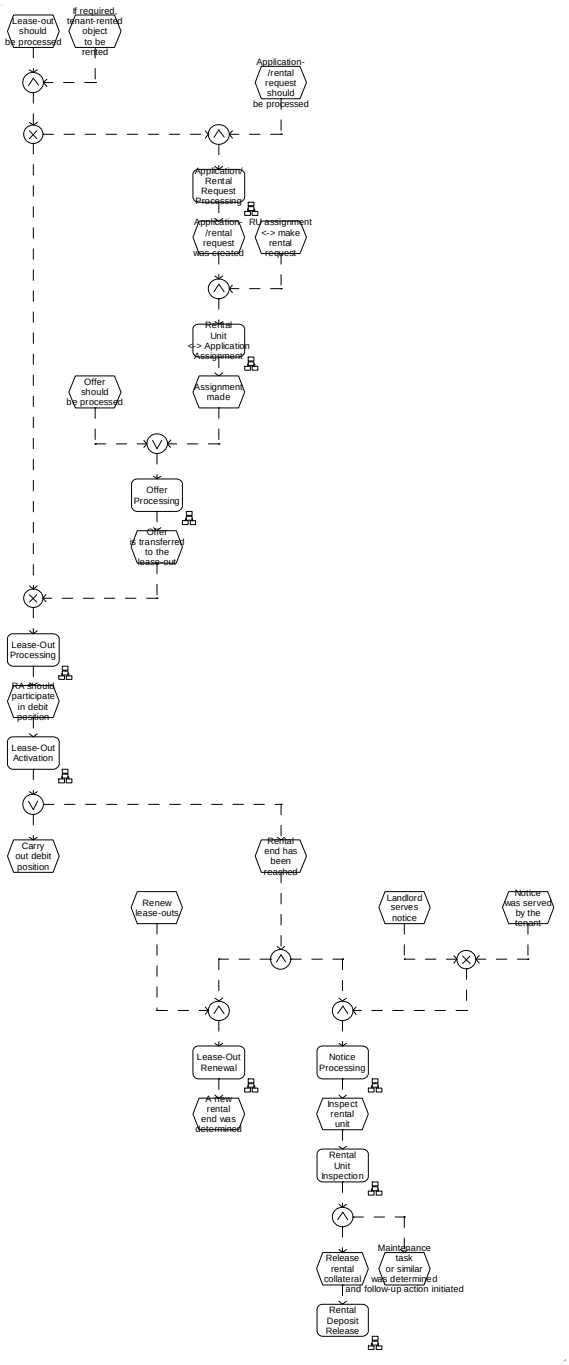


Figure B.46: Real Estate Management – Real Estate Management – Rental (reduced size 16, unsound)

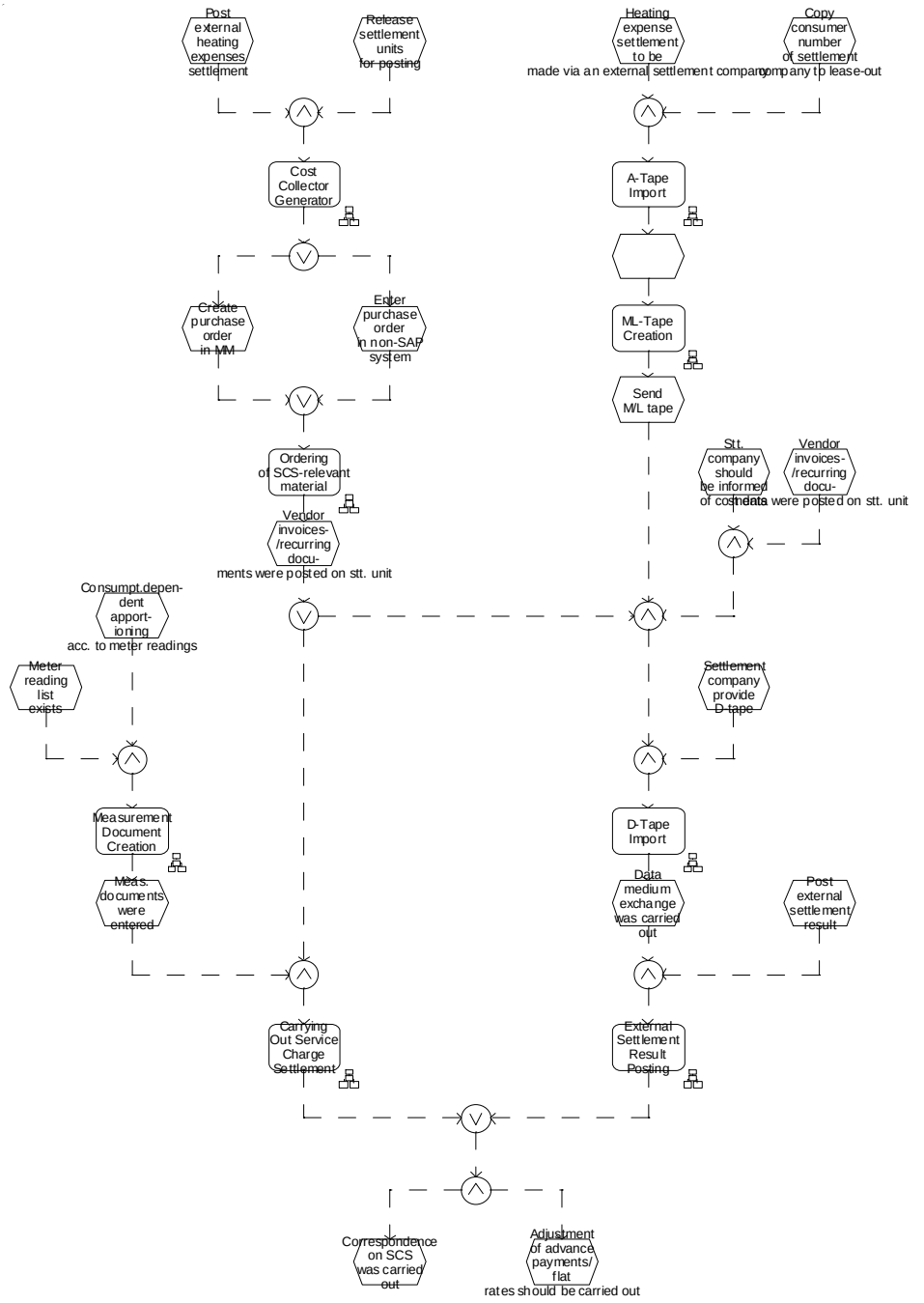


Figure B.47: Real Estate Management – Real Estate Management – Service Charge Settlement (reduced size 8, unsound)

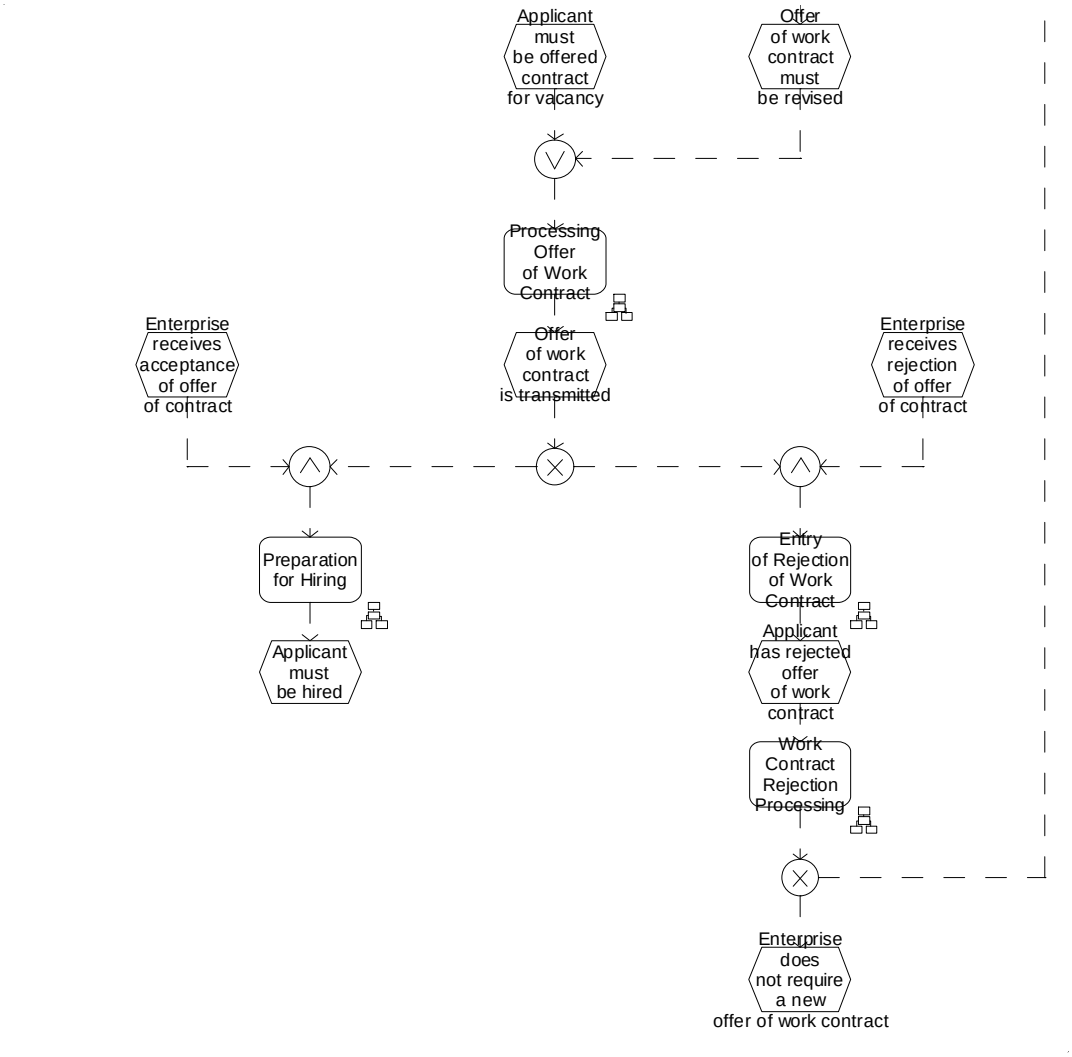


Figure B.48: Recruitment – Recruitment – Work Contract Negotiation (reduced size 10, unsound)

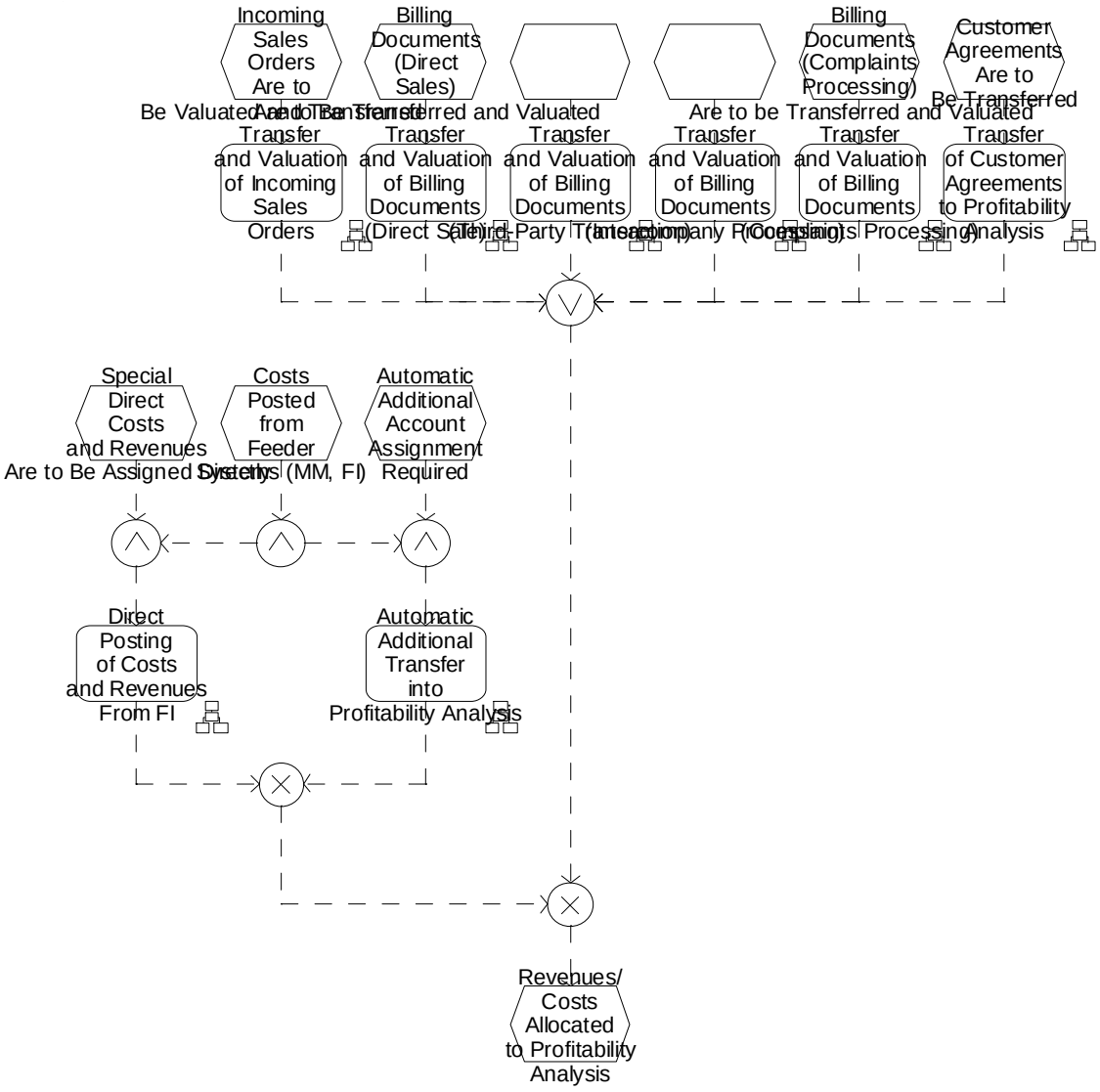


Figure B.49: Revenue and Cost Controlling – Actual Cost/Revenue Allocation – Cost and Revenue Allocation to Profitability Analysis (reduced size 9, unsound)

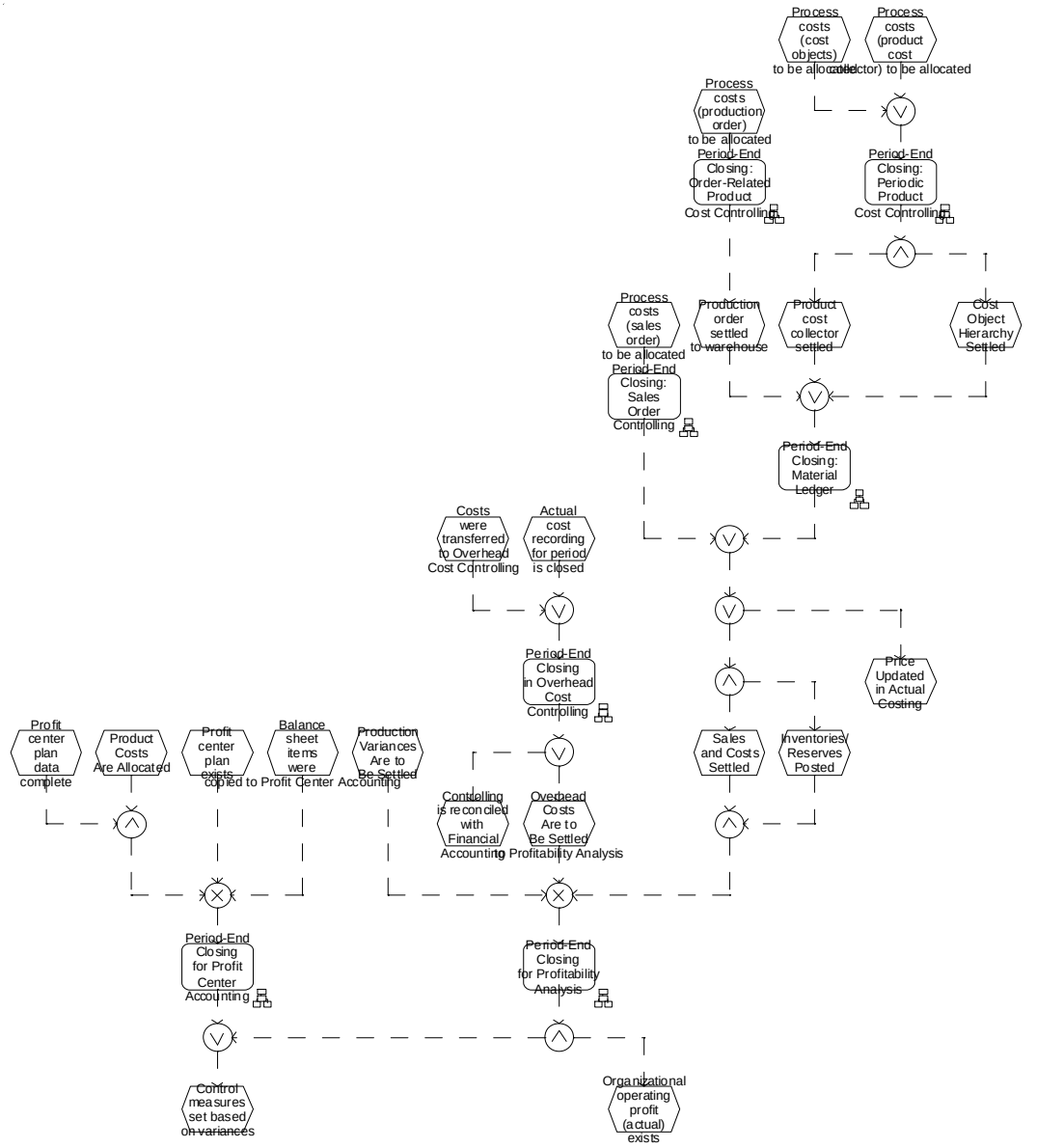


Figure B.50: Revenue and Cost Controlling – Period-End Closing (Controlling) (reduced size 11, sound)

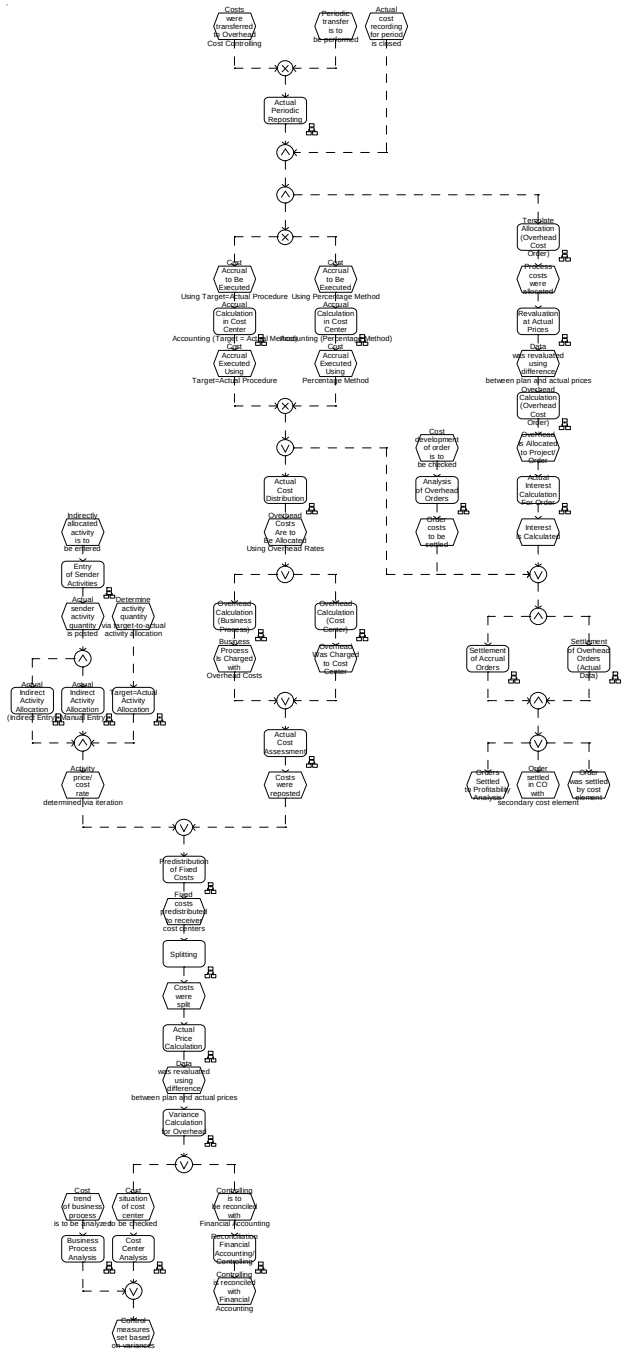


Figure B.51: Revenue and Cost Controlling – Period-End Closing (Controlling) – Period-End Closing in Overhead Cost Controlling (reduced size 13, sound)

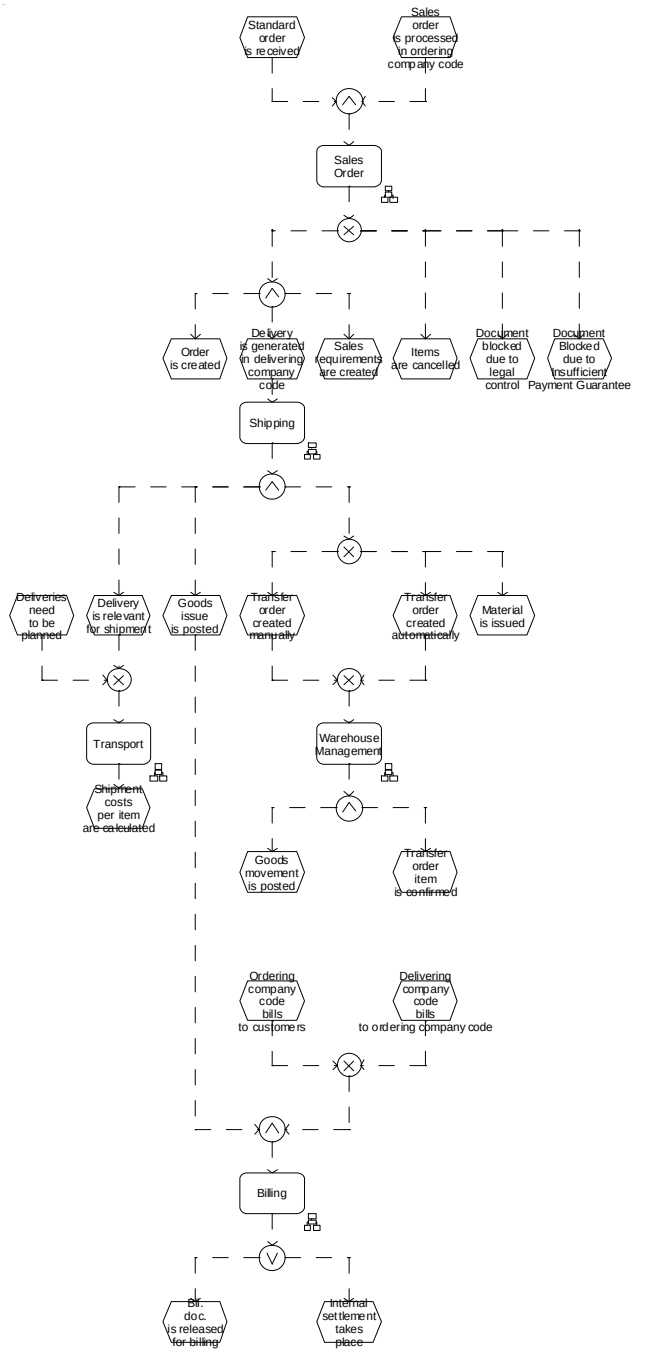


Figure B.52: Sales and Distribution – Intercompany Handling (reduced size 10, unsound)

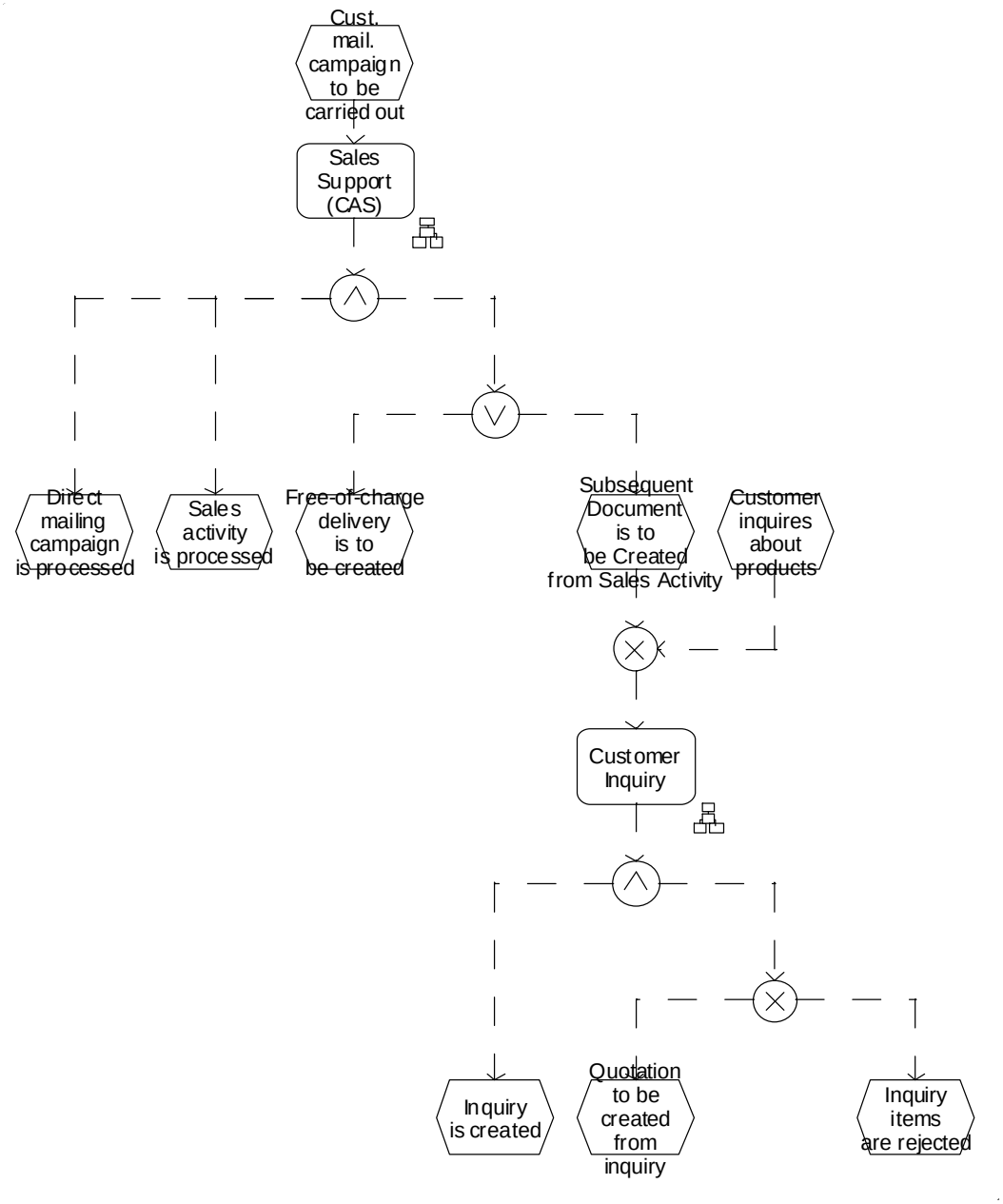


Figure B.53: Sales and Distribution – Pre-Sales Handling (reduced size 6, sound)

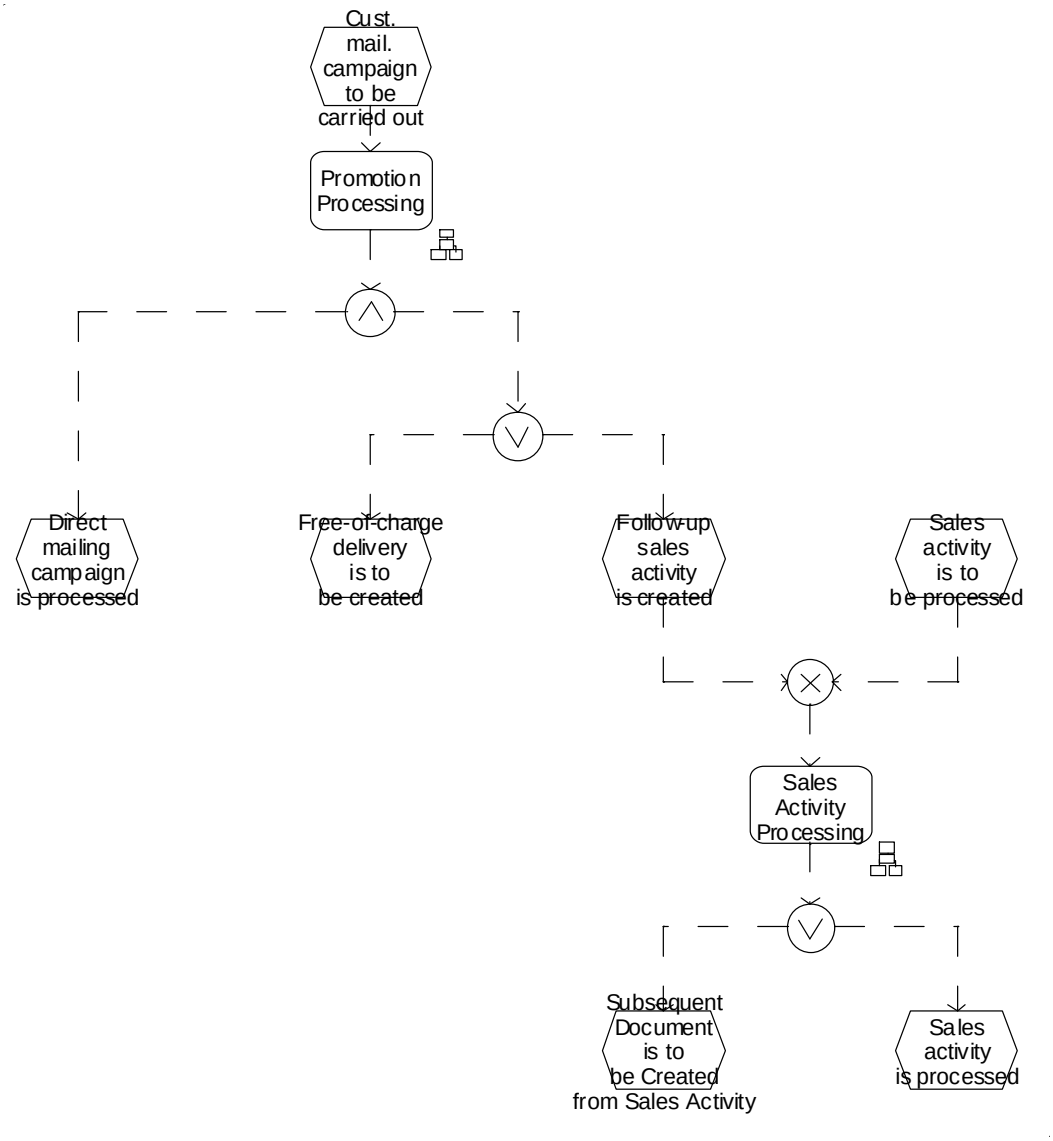


Figure B.54: Sales and Distribution – Pre-Sales Handling – Sales Support (CAS) (reduced size 6, sound)

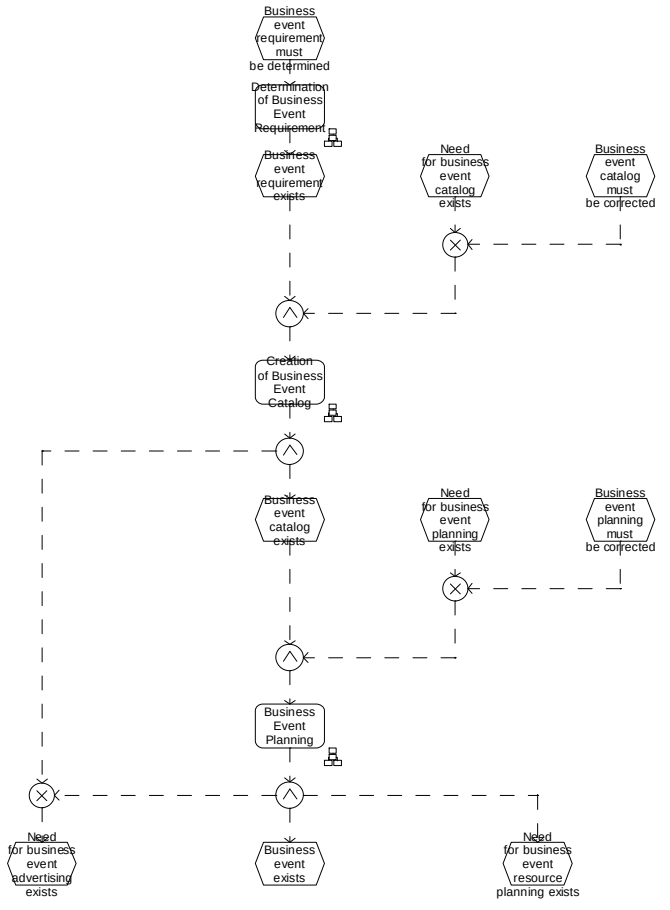


Figure B.55: Training and Event Management – Business Event Planning and Performance – Business Event Planning (reduced size 6, unsound)

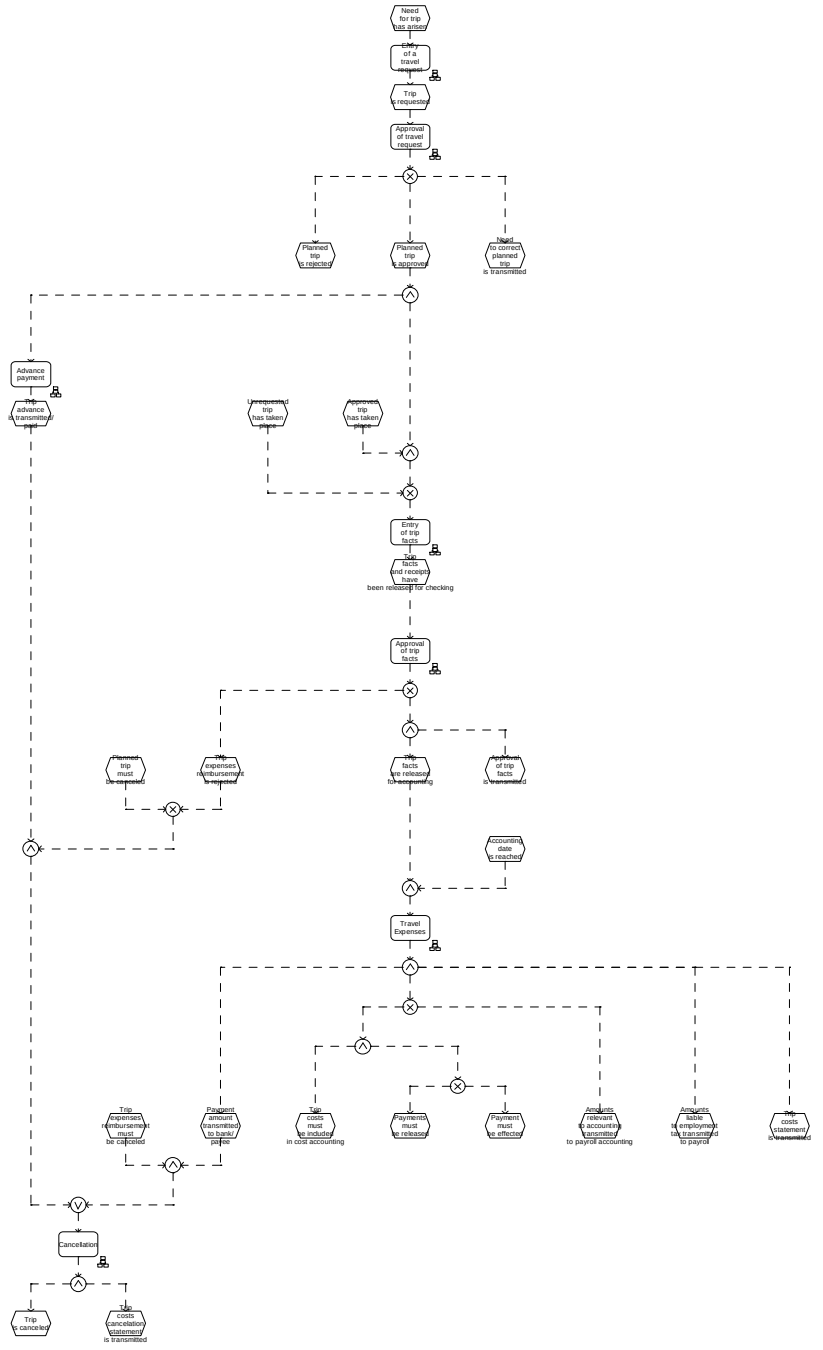


Figure B.56: Travel Management – Travel Expenses (reduced size 16, unsound)

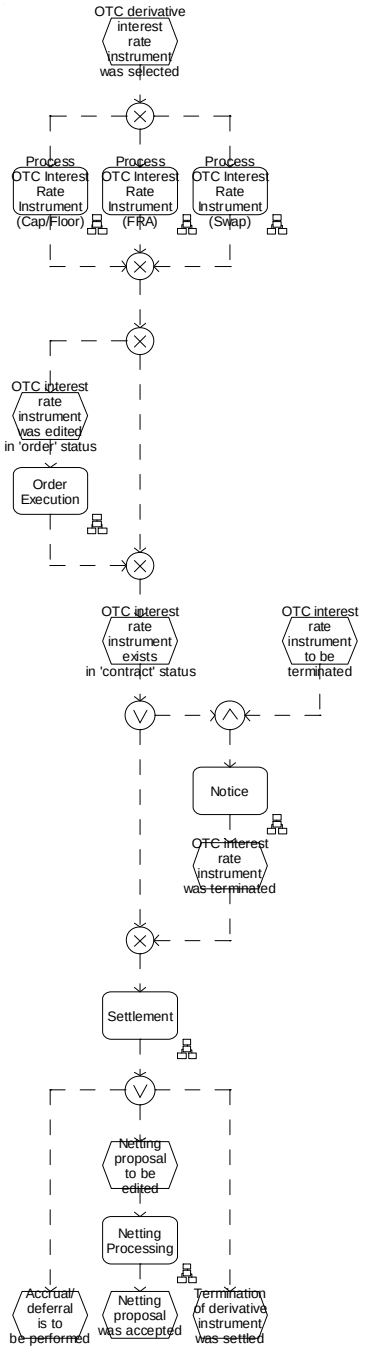


Figure B.57: Treasury – Process OTC Derivative Transactions (TR-DE) – Transaction Processing (reduced size 6, unsound)

Appendix C

Descriptive Statistics of Variables

This appendix gathers details of the statistical analysis. In particular, Section C.1 gives a tabular overview of the variables that were available for the statistical analysis. Section C.2 presents box plots that illustrate the empirical distribution of the variables disaggregated by the group of models. Section C.3 shows box plots for the different variables disaggregated by the variable *hasErrors*. Finally, Section C.5 contains the correlation tables between the variable *hasError* and the different metrics.

C.1 Definition of Variables

This section gives two tables that describe the variables that were available for the statistical analysis. Apart from the variable *countProM* and *hasErrors* all variable values were generated by *xoEPC*.

Table C.1: Variables of the analysis table (first part)

Variable name	**Description**
Group	Number of the EPC collection group
Filename	Name of the ARIS XML file
Model ID	ID of the EPC model
Duration	Processing time in milliseconds
Path	Path of the EPC within the model hierarchy of the file
Name	Name of the EPC model
Error	Value 1 if xoEPC found errors, otherwise 0
Reduced	Value 1 if the EPC was reduced completely, otherwise 0
Restsize	Size in nodes of the reduced EPC
Interpretable	Value 1 if relaxed syntactically correct, otherwise 0
Syntax	List of syntax error descriptions
N	Number of nodes
C	Number of connectors
E	Number of events
Es	Number of start events
Ee	Number of end events
F	Number of functions
AND	Number of AND-connectors
XOR	Number of XOR-connectors
OR	Number of OR-connectors
ANDj	Number of AND-joins
XORj	Number of XOR-joins
ORj	Number of OR-joins
ANDs	Number of AND-splits
XORs	Number of XOR-splits
ORs	Number of OR-splits
A	Number of arcs
diameter	Diameter

Table C.2: Variables of the analysis table (second part)

Variable name	Description
Density	Density metric
CNC	Coefficient of connectivity
AvCDegree	Average connector degree
MaxCDegree	Maximum connector degree
Separability	Separability ratio
Sequentiality	Sequentiality ratio
Structuredness	Structuredness ratio
Depth	Depth
MM	Connector mismatch
cHeterogeneity	Connector heterogeneity
CFC	Control flow complexity
CYC	Cyclicity
tokenSplit	Token split
rsequence	Number of trivial construct rule application
rblock	Number of structured block rule application
rloop	Number of structured loop rule application
rstartend	Number of structured start and end rule application
rjump	Number of unstructured start and end rule application
rdelta	Number of delta rule application
rprism	Number of prism rule application
rmerge	Number of merge rule application
rxoronly	Number of nodes deleted by homogeneous rule application
countblock	Number of structured block errors
countloop	Number of structured loop errors
countdelta	Number of delta errors
countprism	Number of prism errors
countsplitend	Number of unstructured start and end errors
countProM	Value 1 if errors detected by ProM, otherwise 0
hasErrors	Value 1 if errors, otherwise 0

C.2 Box plots filtered by model group

This section shows box plots of each variable disaggregated by the group of models. The boolean variables *Error*, *Reduced*, *Interpretable*, *countProM*, and *hasError* are not included since box plots are made for interval scale.

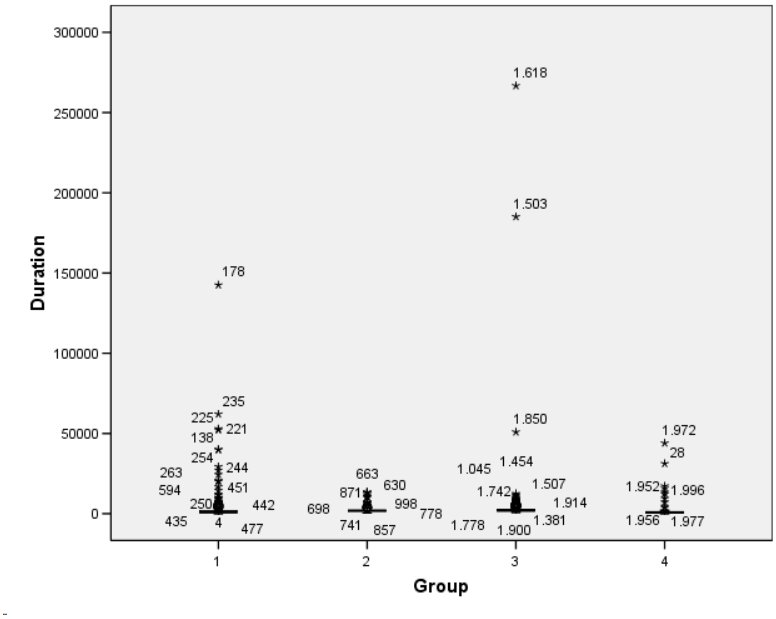


Figure C.1: Box plot for duration by group

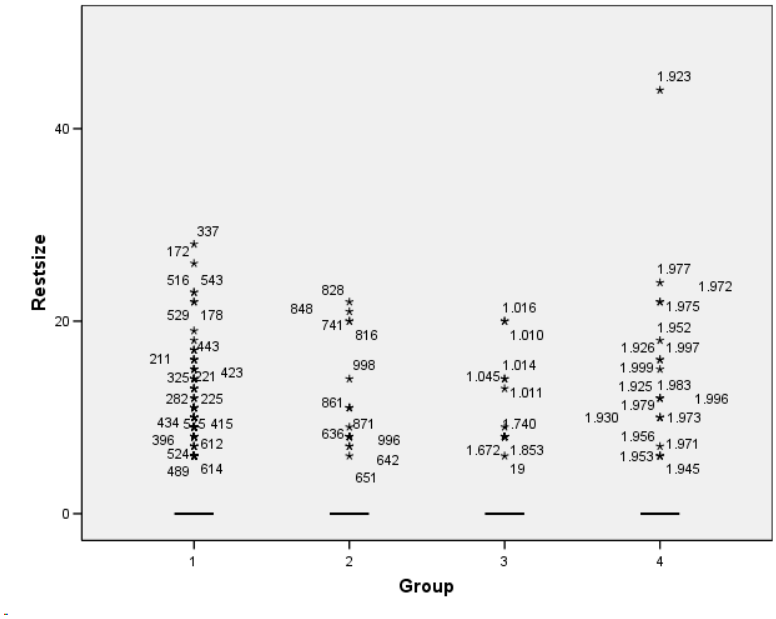


Figure C.2: Box plot for restsize by group

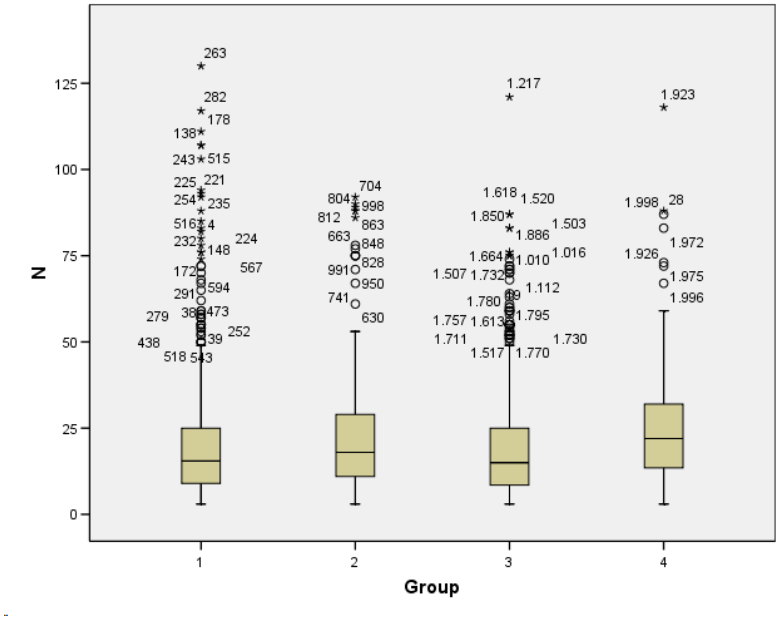


Figure C.3: Box plot for nodes N by group

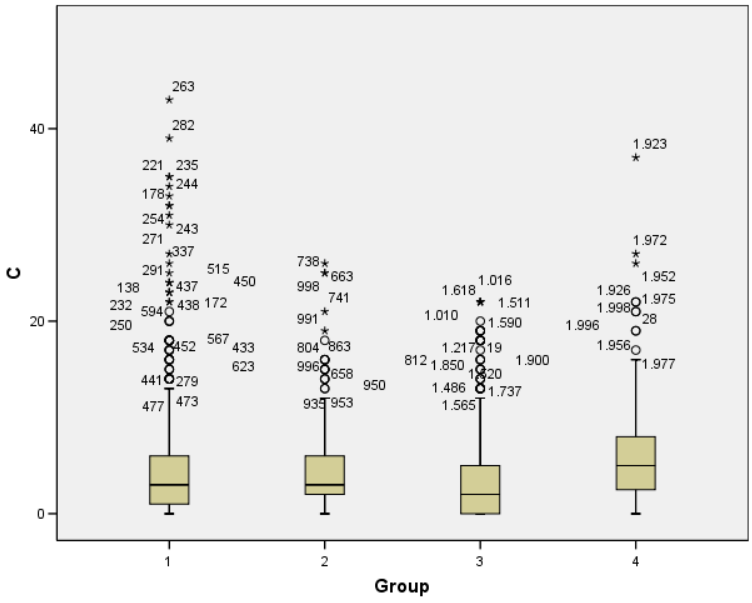


Figure C.4: Box plot for connectors C by group

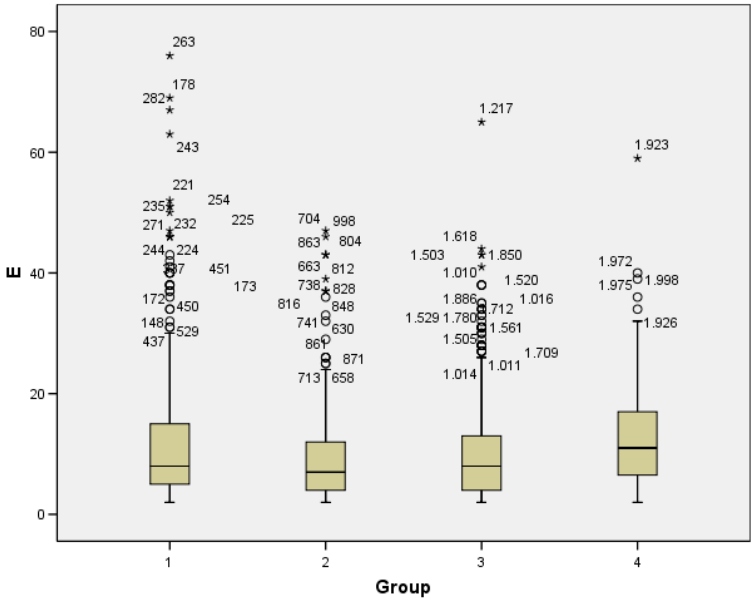


Figure C.5: Box plot for events E by group

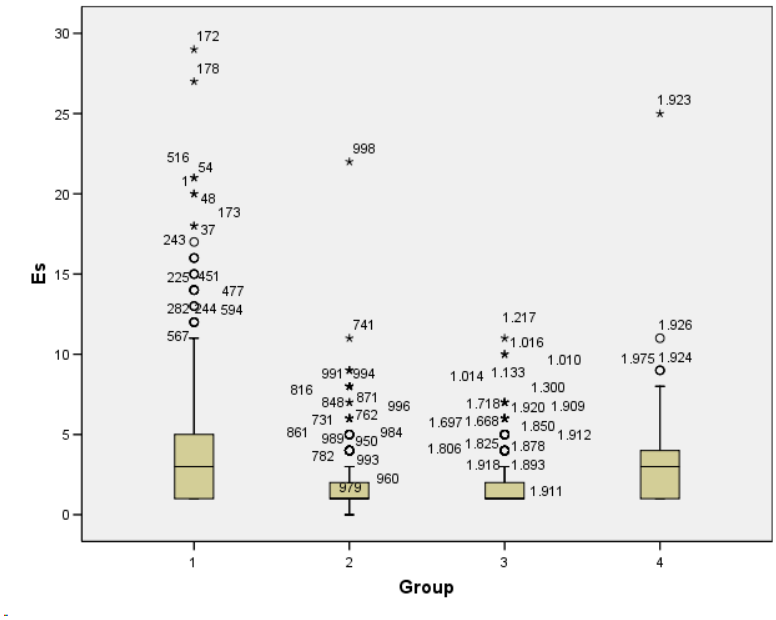


Figure C.6: Box plot for start events Es by group

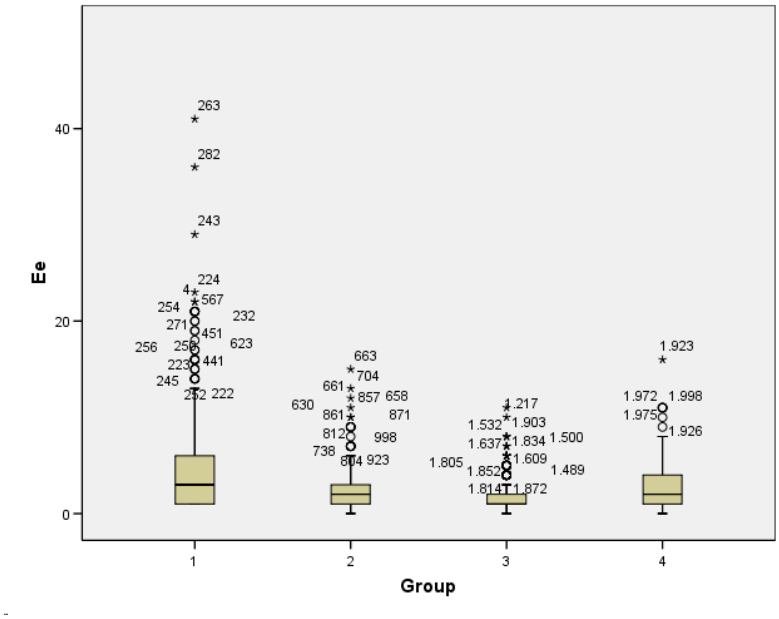


Figure C.7: Box plot for end events Ee by group

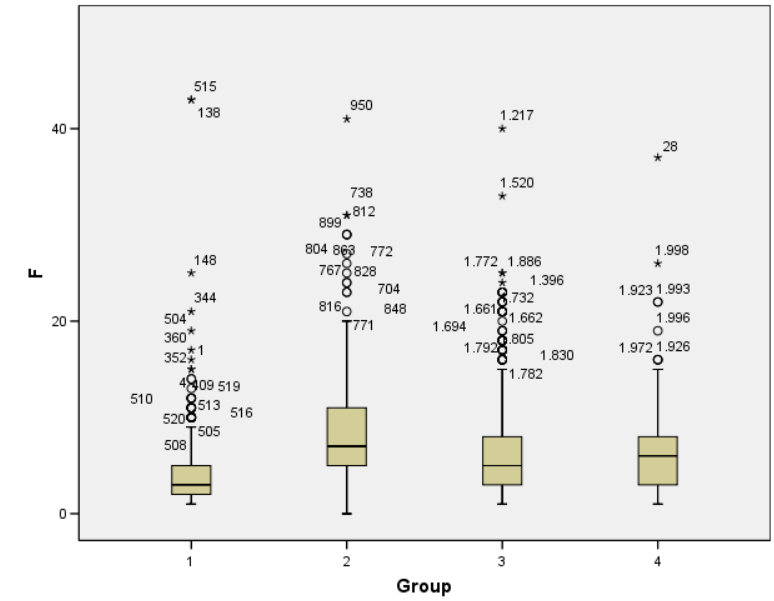


Figure C.8: Box plot for functions F by group

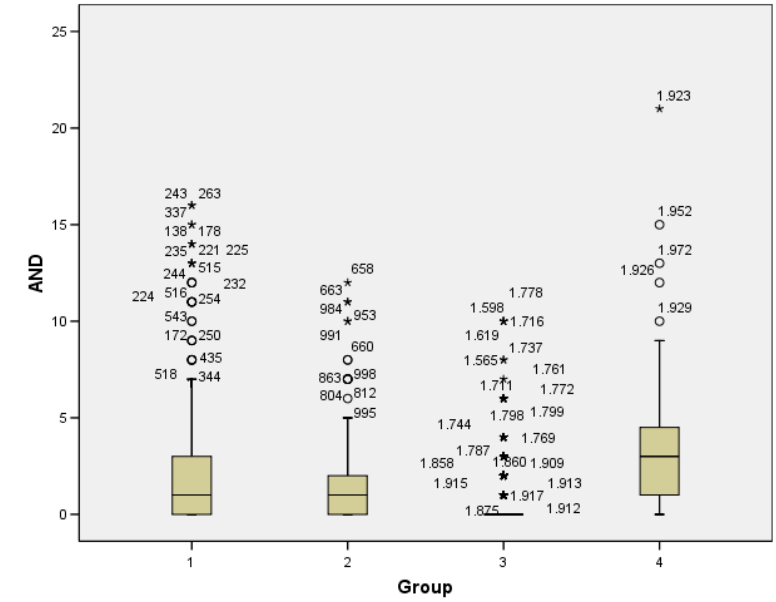


Figure C.9: Box plot for AND-connectors by group

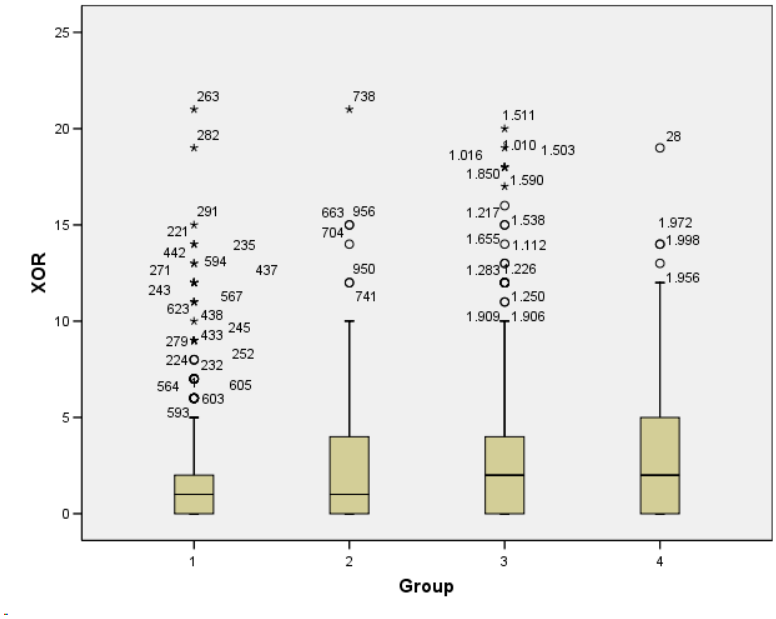


Figure C.10: Box plot for XOR-connectors by group

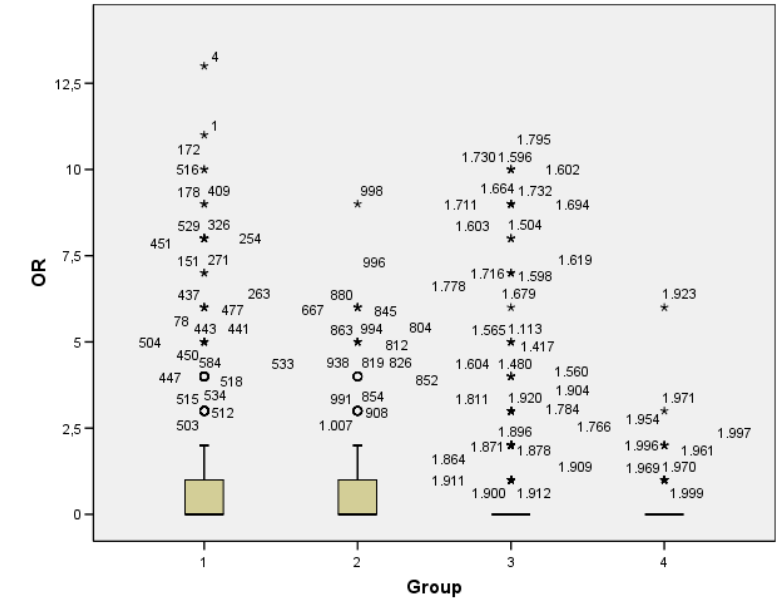


Figure C.11: Box plot for OR-connectors by group

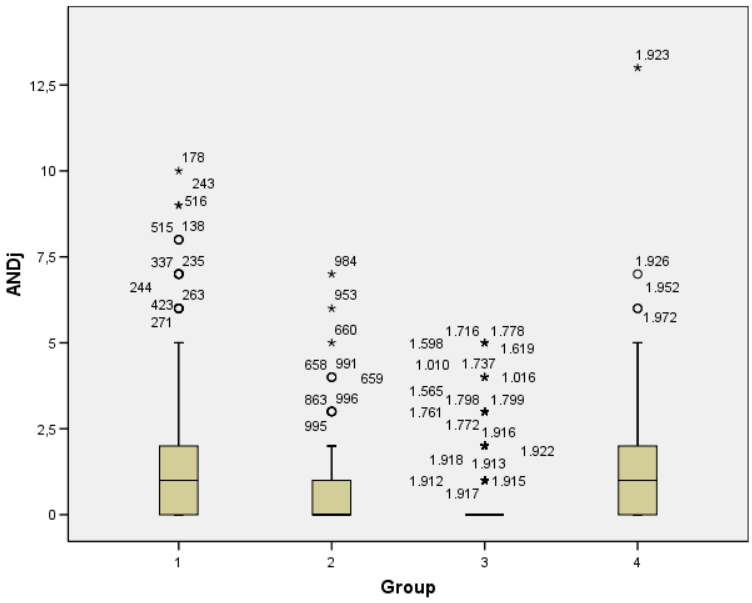


Figure C.12: Box plot for AND-joins by group

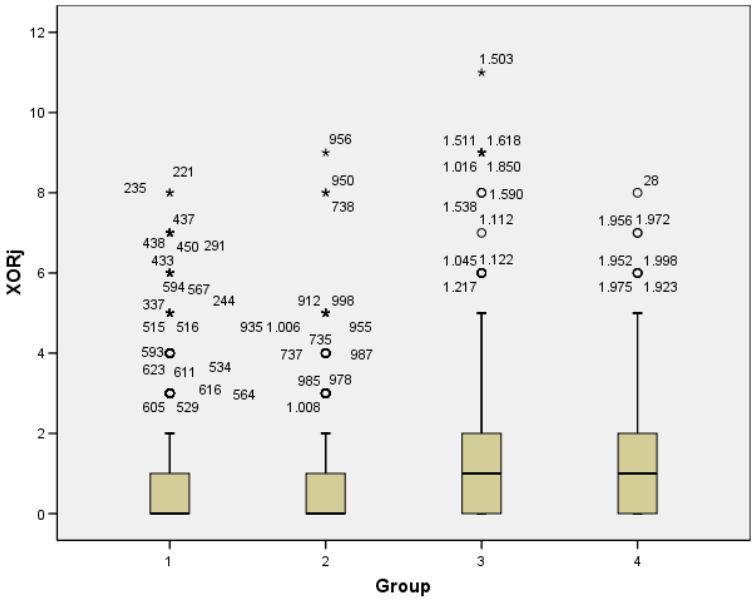


Figure C.13: Box plot for XOR-joins by group

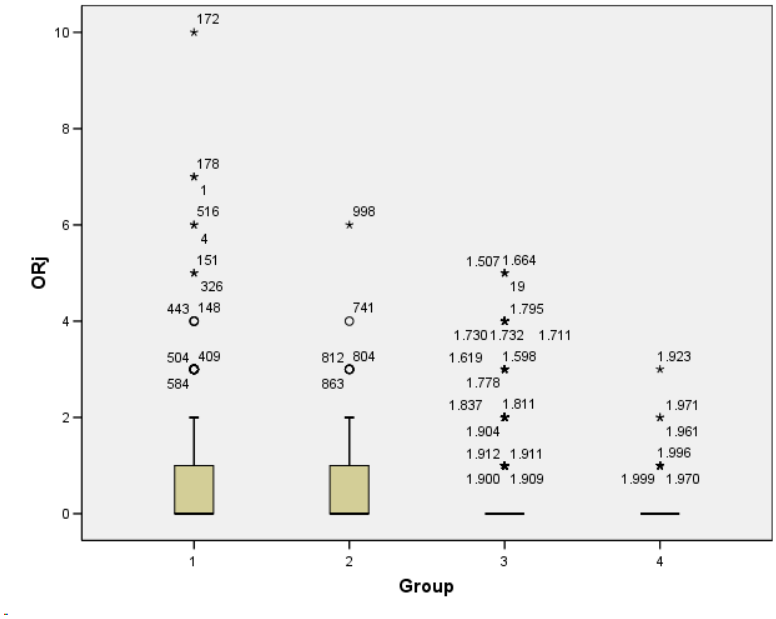


Figure C.14: Box plot for OR-joins by group

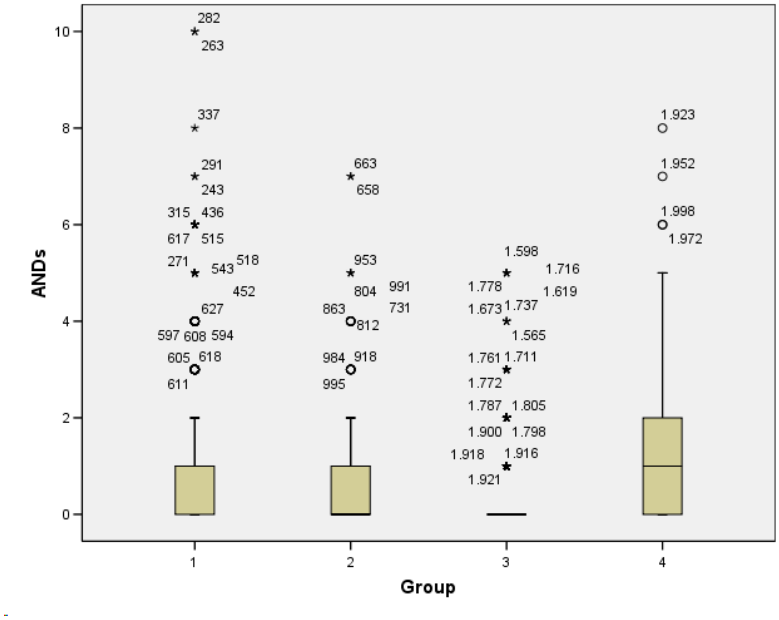


Figure C.15: Box plot for AND-splits by group

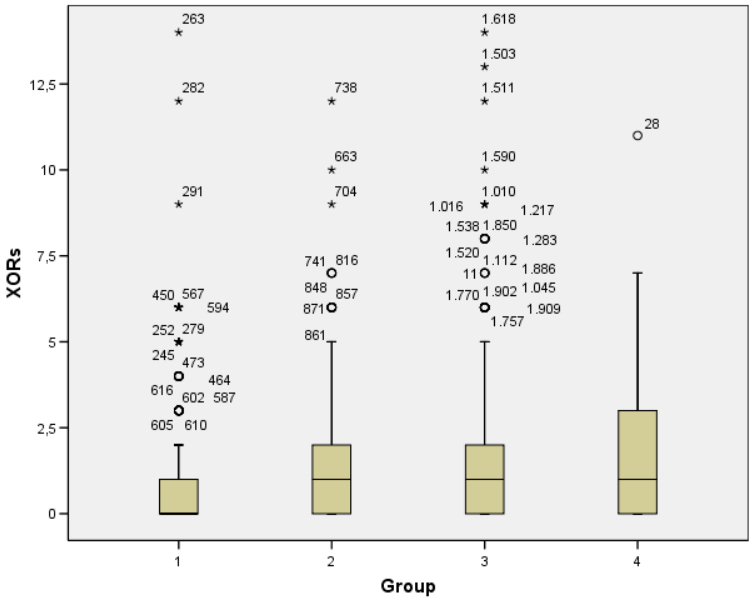


Figure C.16: Box plot for XOR-splits by group

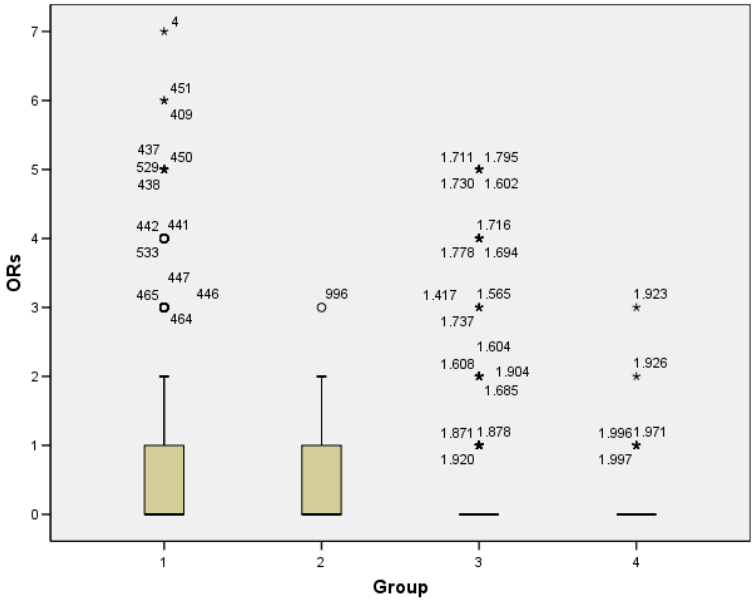


Figure C.17: Box plot for OR-splits by group

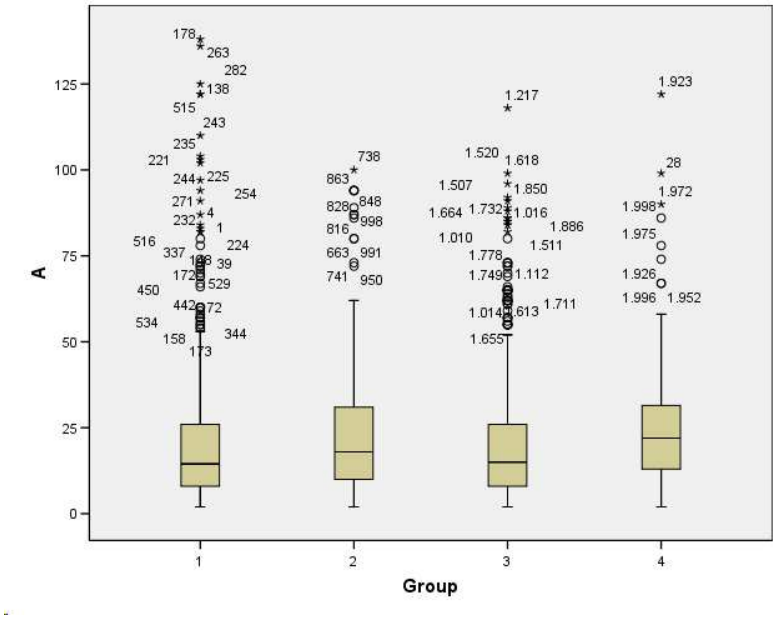


Figure C.18: Box plot for arcs A by group

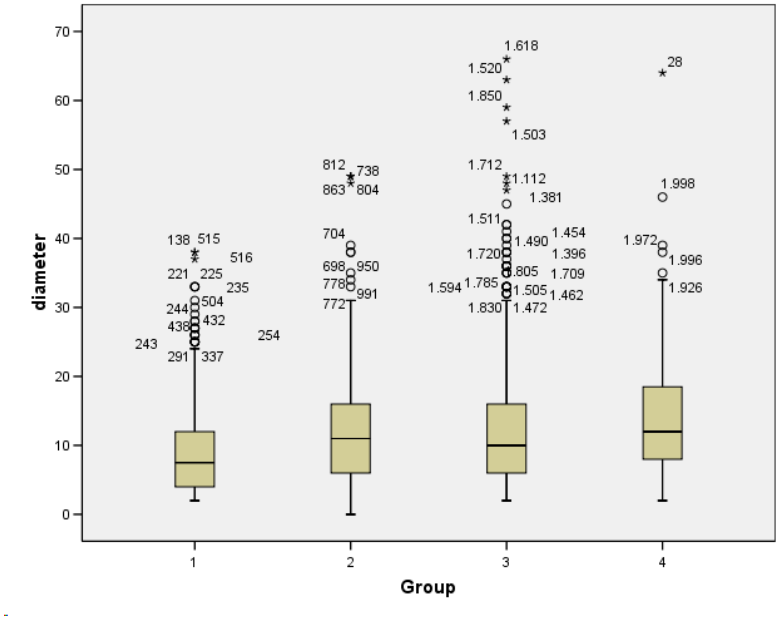


Figure C.19: Box plot for diameter by group

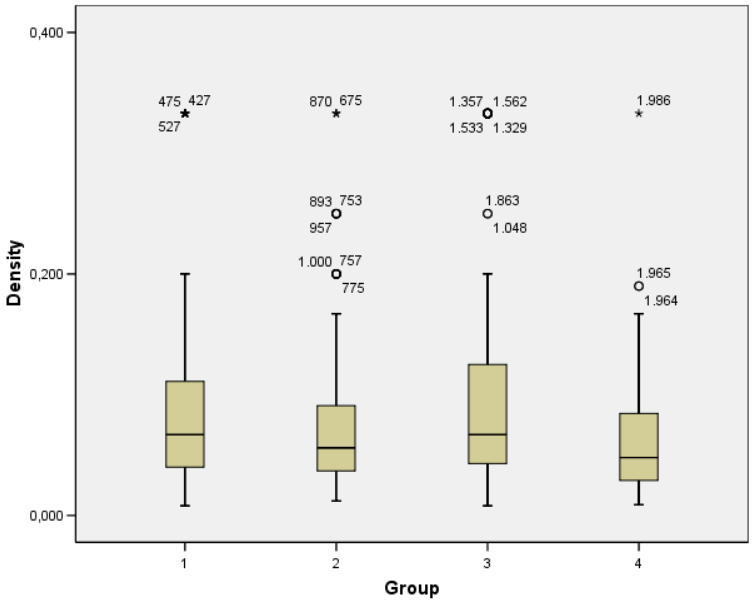


Figure C.20: Box plot for density by group

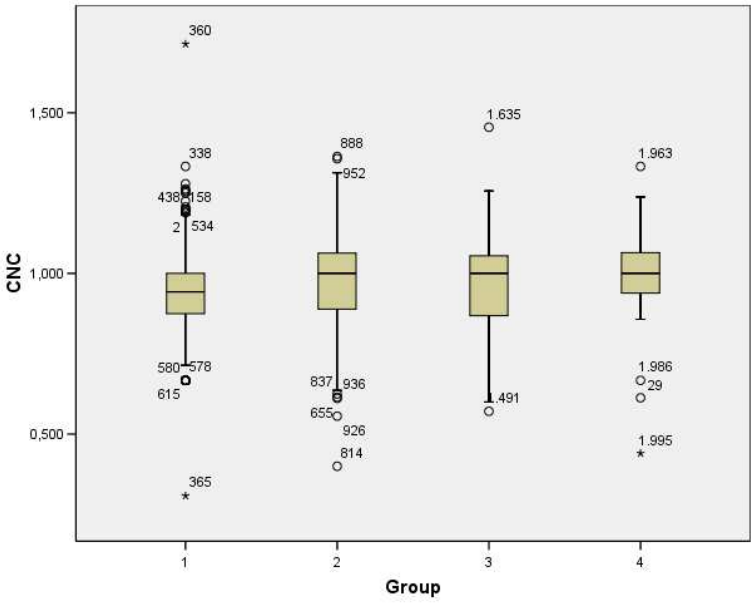


Figure C.21: Box plot for coefficient of connectivity CNC by group

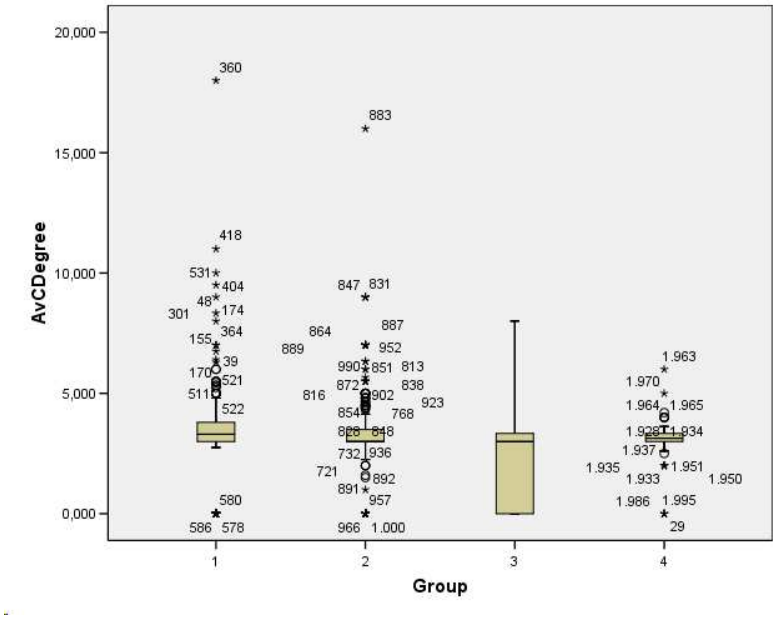


Figure C.22: Box plot for average connector degree by group

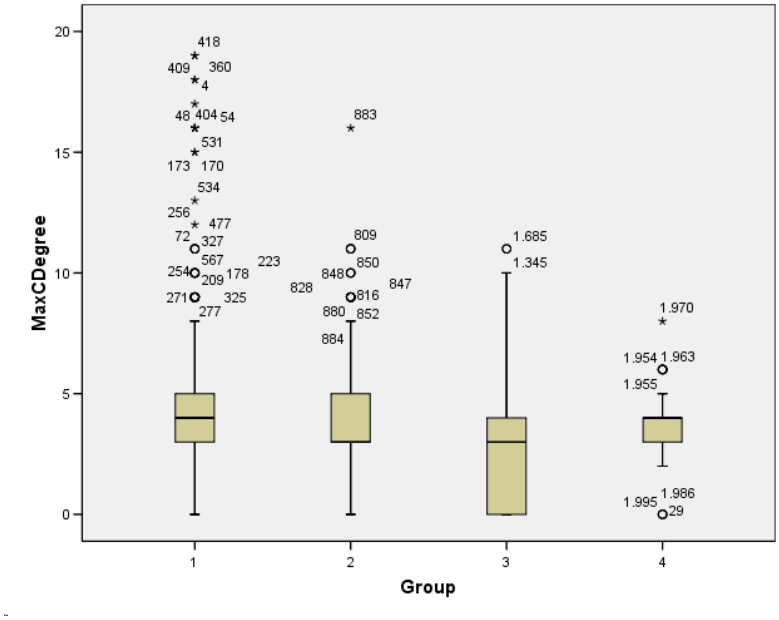


Figure C.23: Box plot for maximum connector degree by group

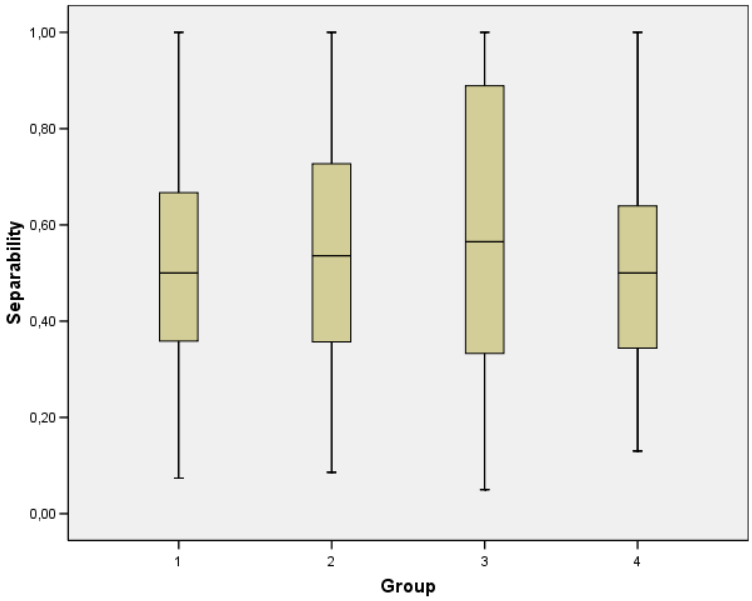


Figure C.24: Box plot for separability by group

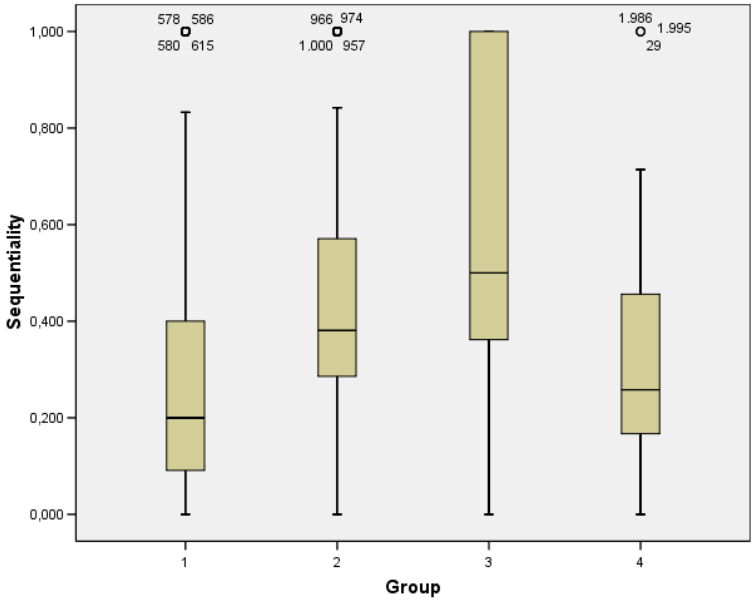


Figure C.25: Box plot for sequentiality by group

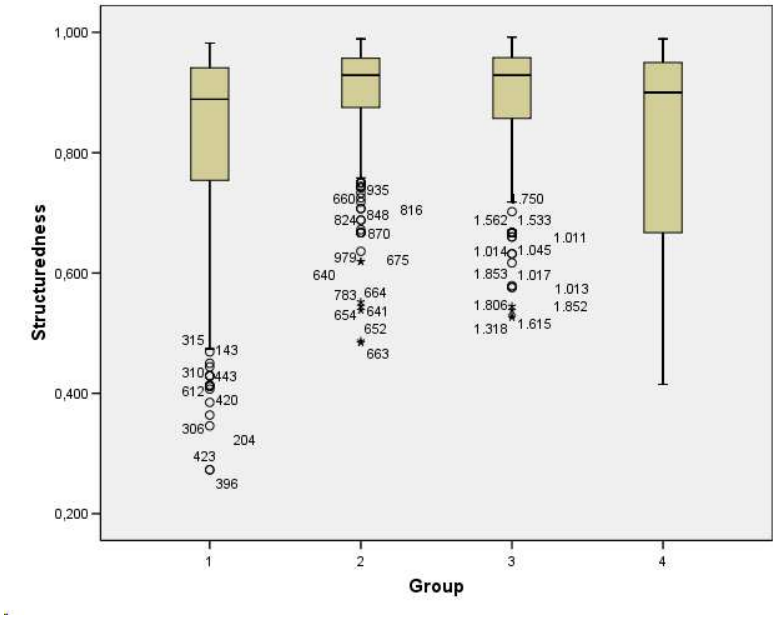


Figure C.26: Box plot for structuredness by group

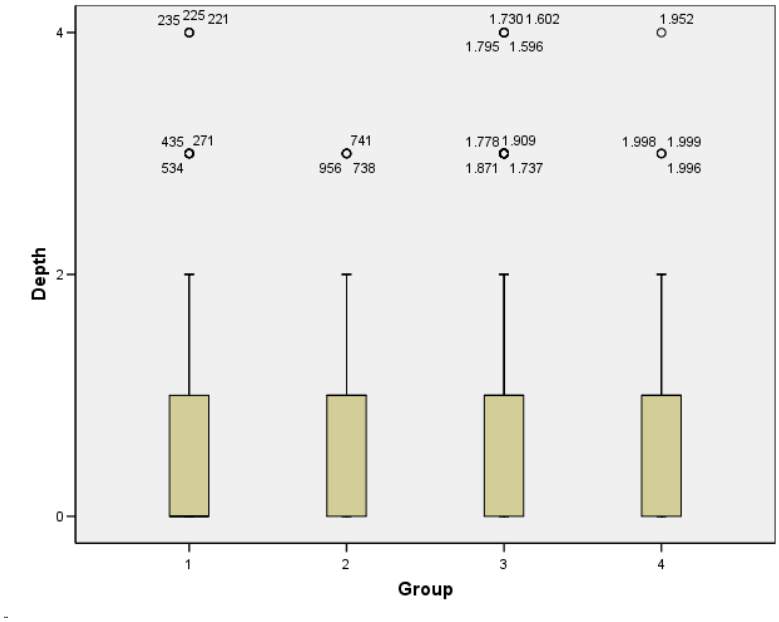


Figure C.27: Box plot for depth by group

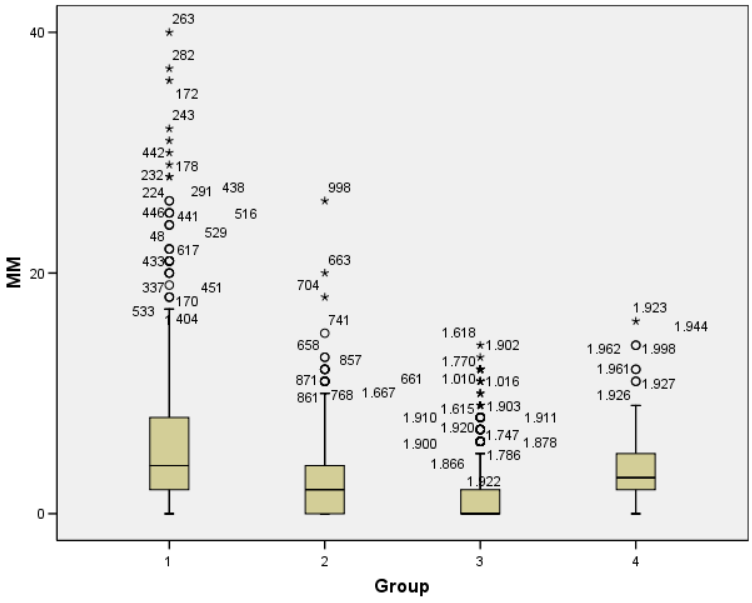


Figure C.28: Box plot for connector mismatch MM by group

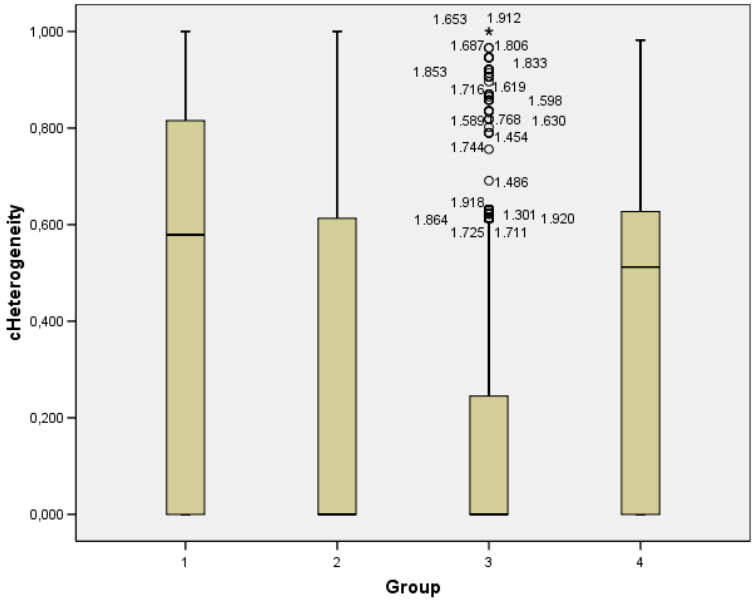


Figure C.29: Box plot for connector heterogeneity by group

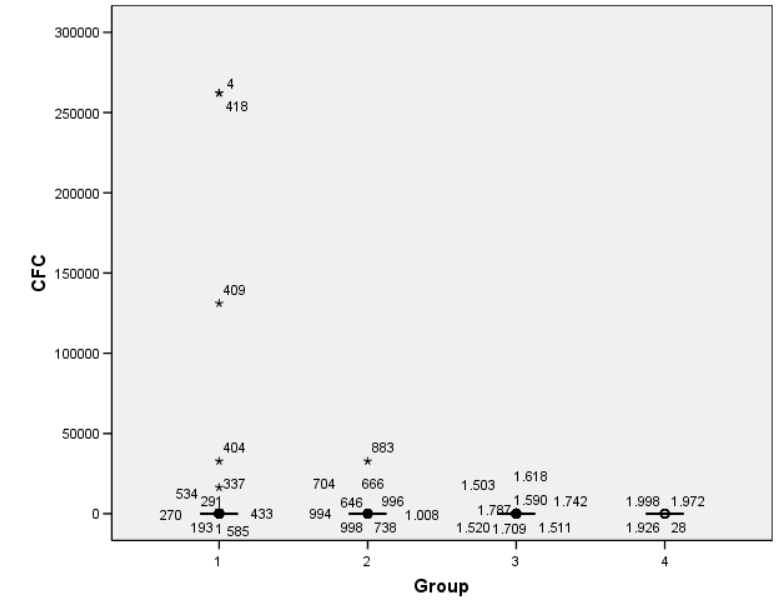


Figure C.30: Box plot for control flow complexity CFC by group

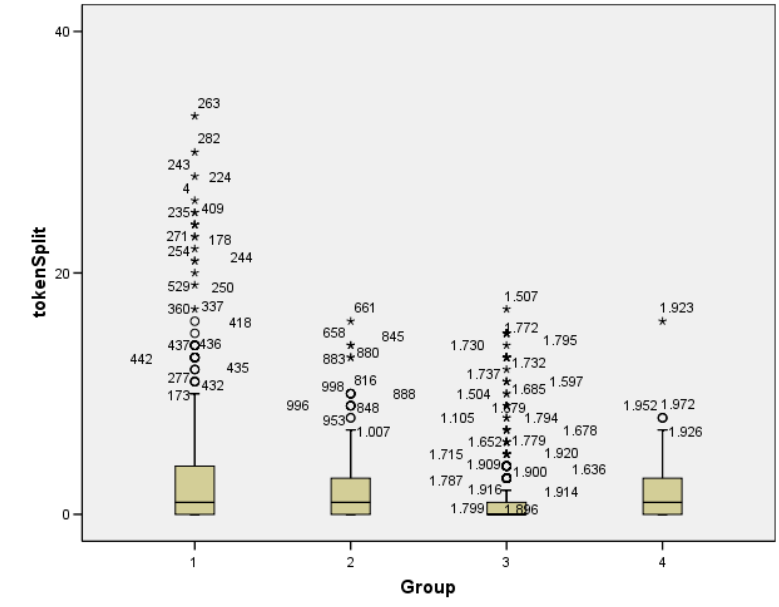


Figure C.31: Box plot for token split by group

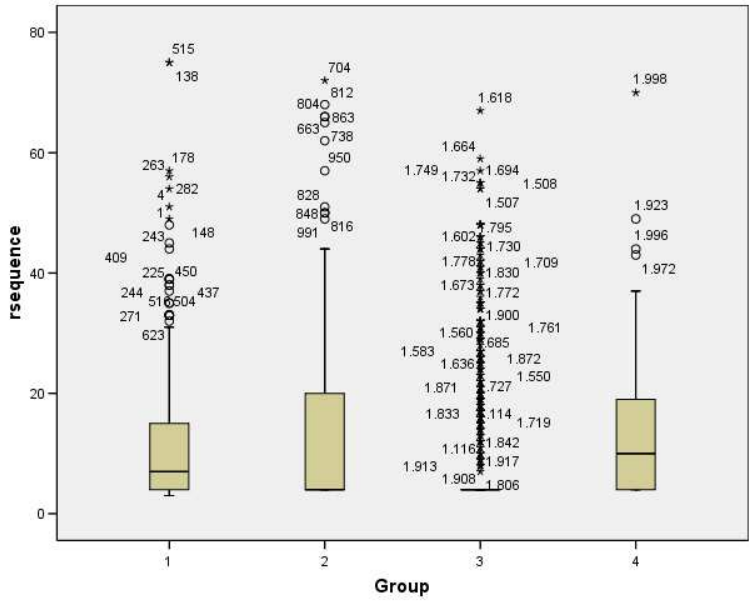


Figure C.32: Box plot for trivial construct rule application by group

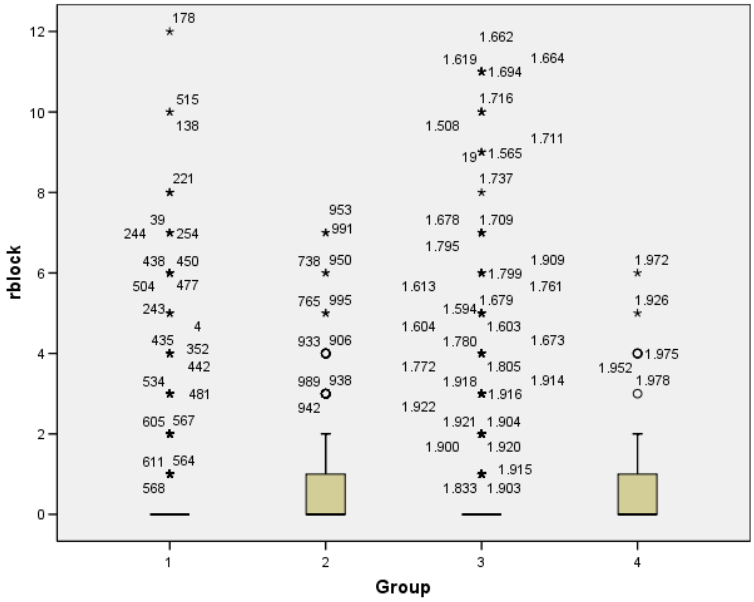


Figure C.33: Box plot for structured block rule application by group

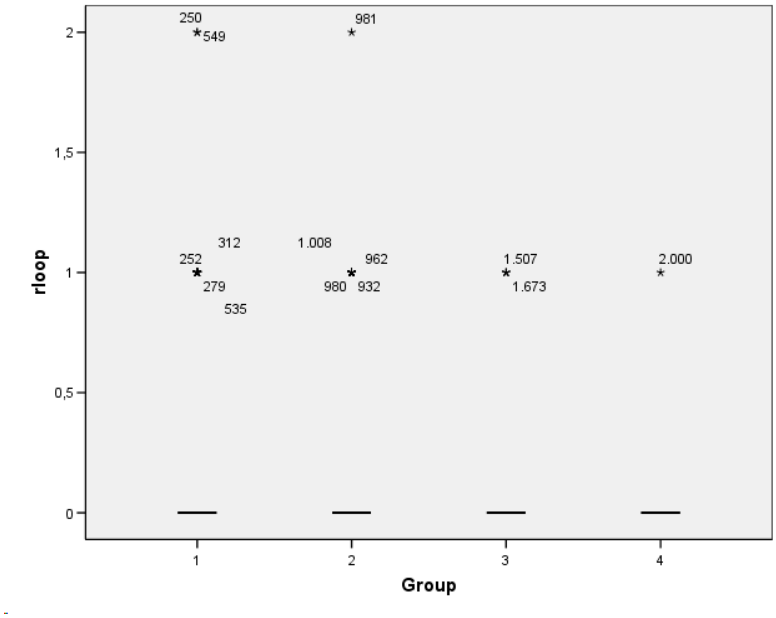


Figure C.34: Box plot for structured loop rule application by group

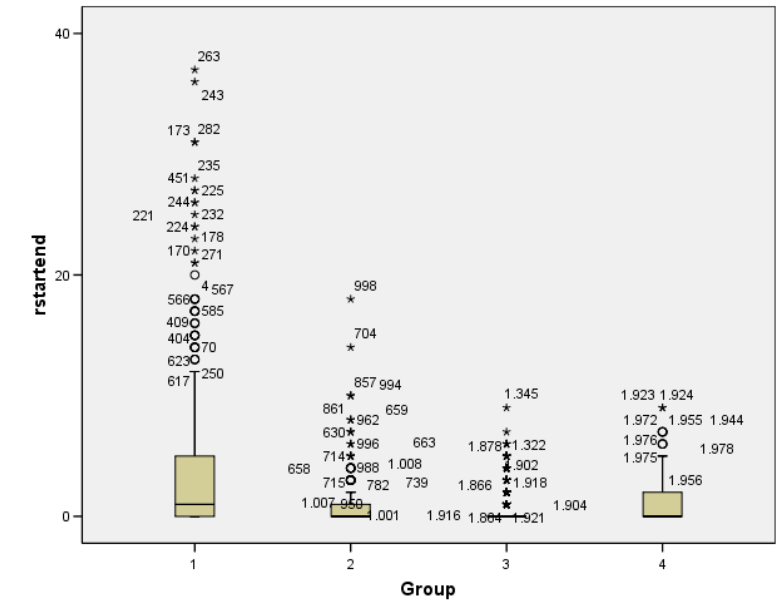


Figure C.35: Box plot for structured start and end rule application by group

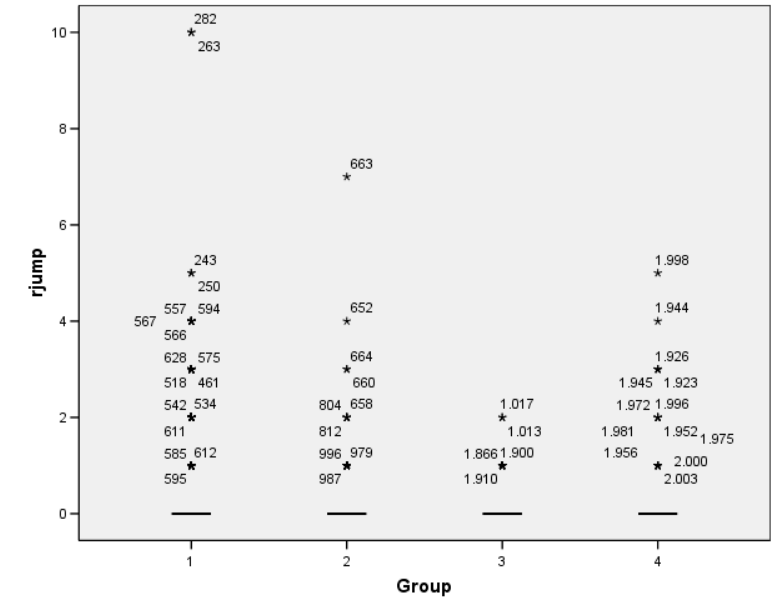


Figure C.36: Box plot for unstructured start and end rule application by group

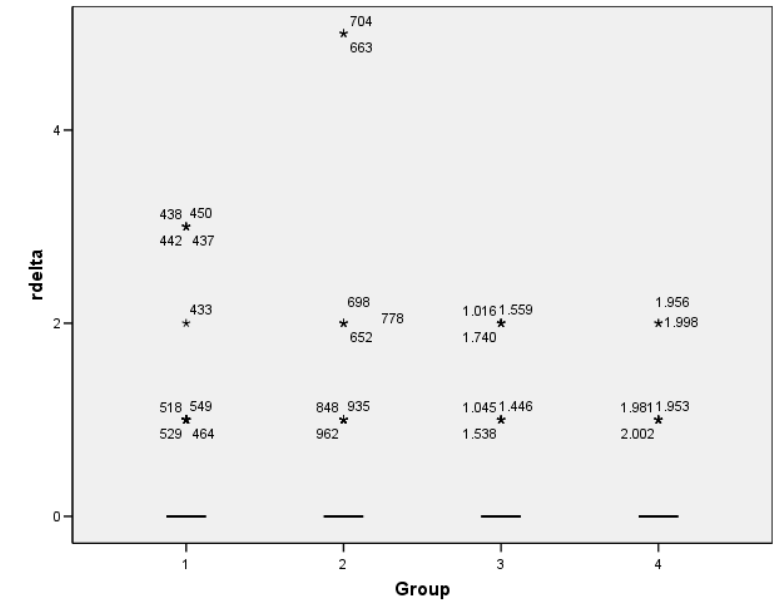


Figure C.37: Box plot for delta rule application by group

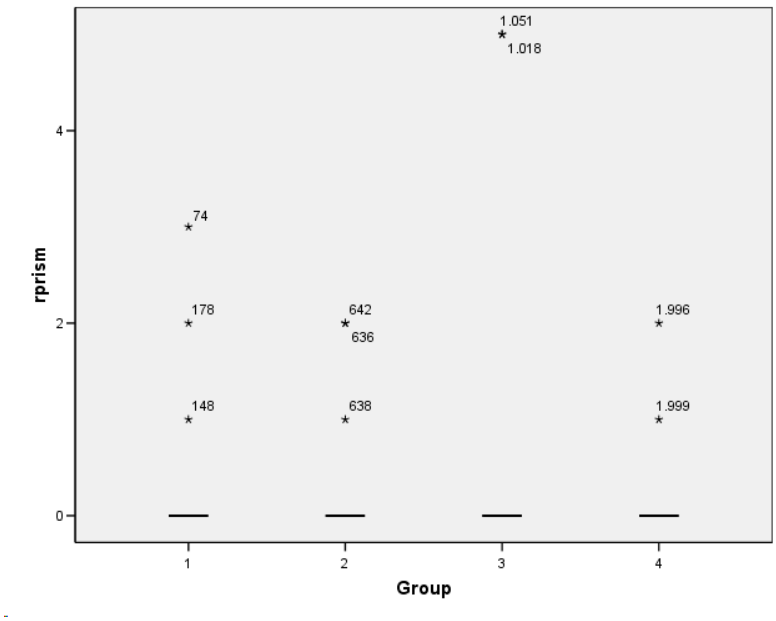


Figure C.38: Box plot for prism rule application by group

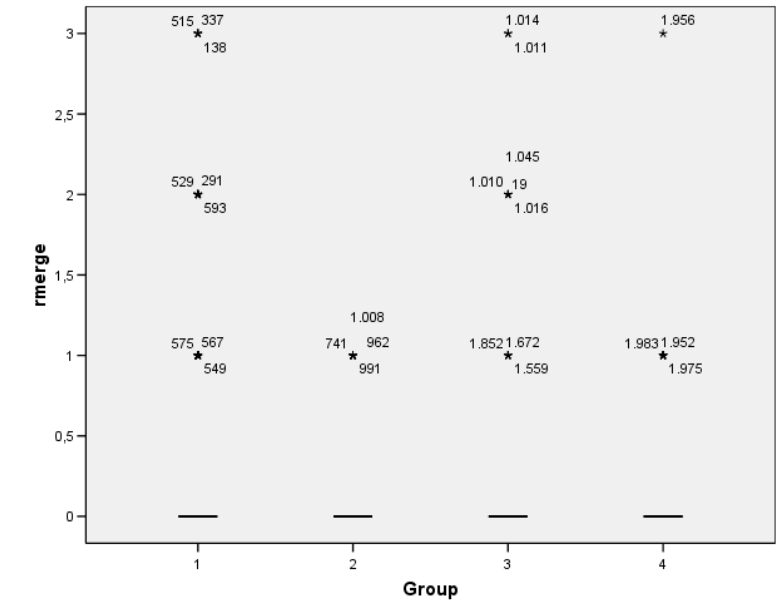


Figure C.39: Box plot for connector merge rule application by group

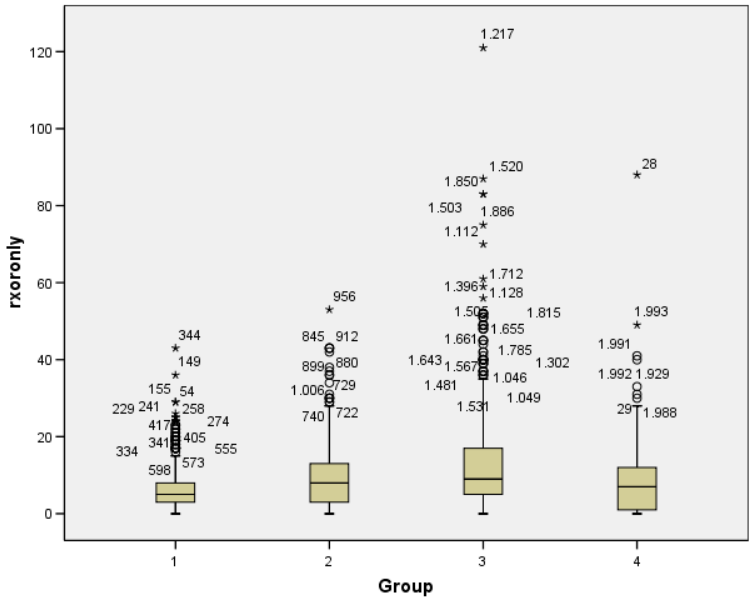


Figure C.40: Box plot for homogeneous rule application by group

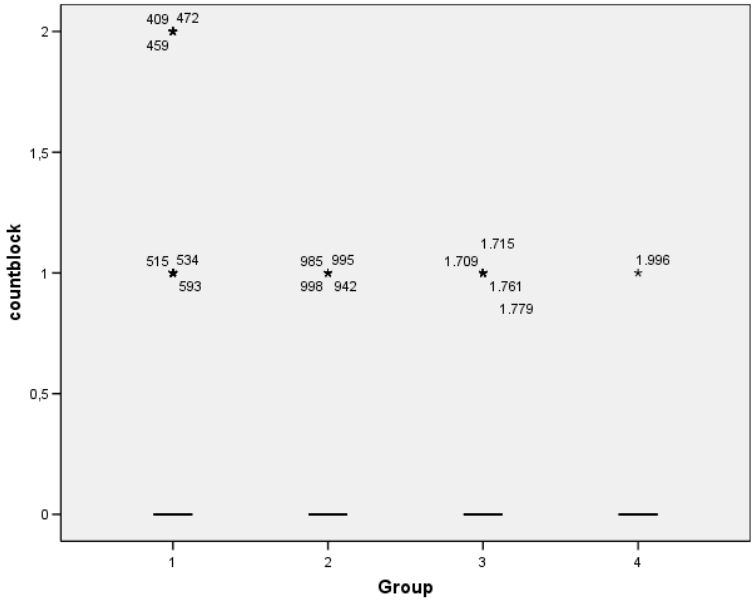


Figure C.41: Box plot for structured block errors by group

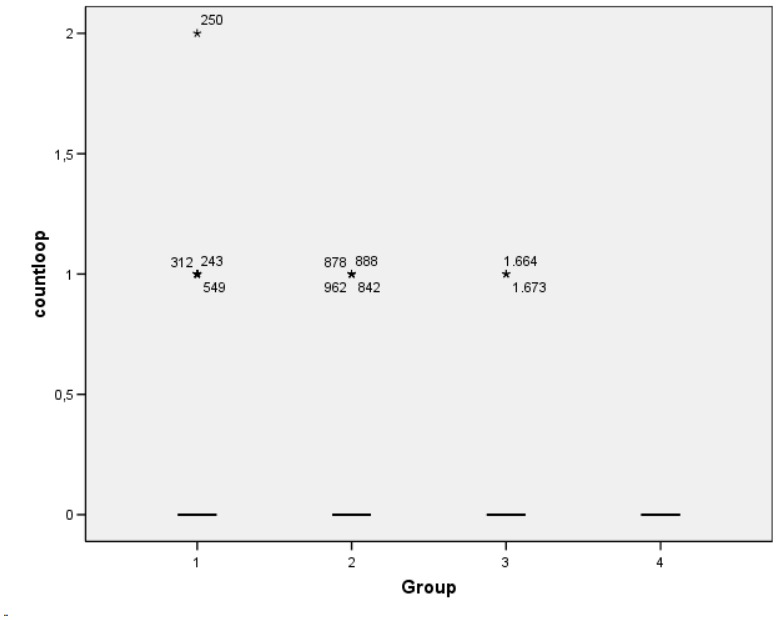


Figure C.42: Box plot for structured loop errors by group

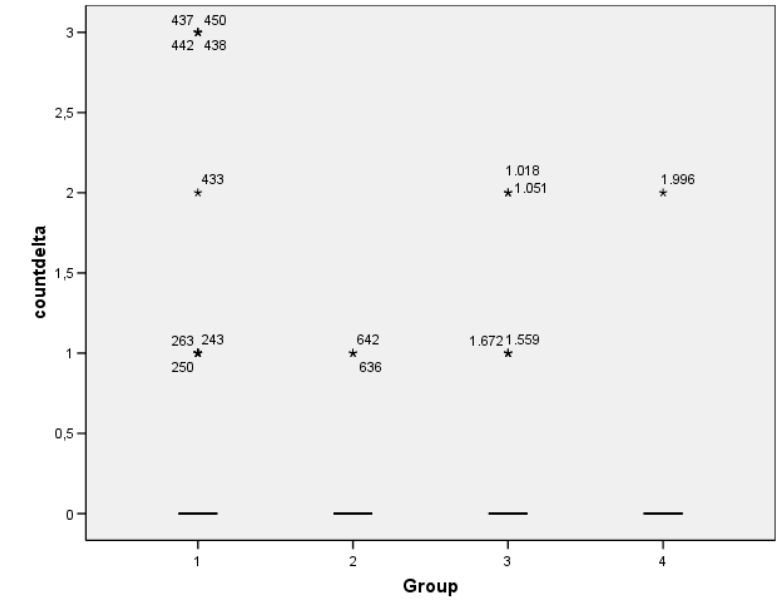


Figure C.43: Box plot for delta errors by group

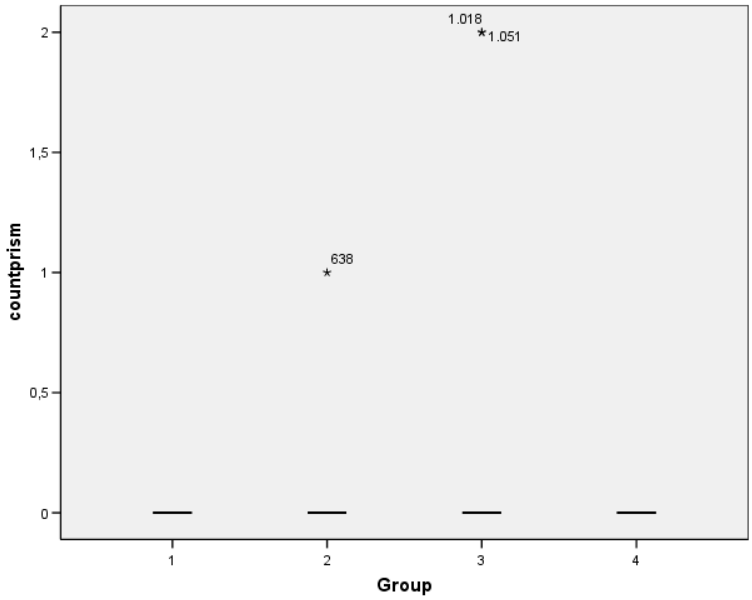


Figure C.44: Box plot for prism errors by group

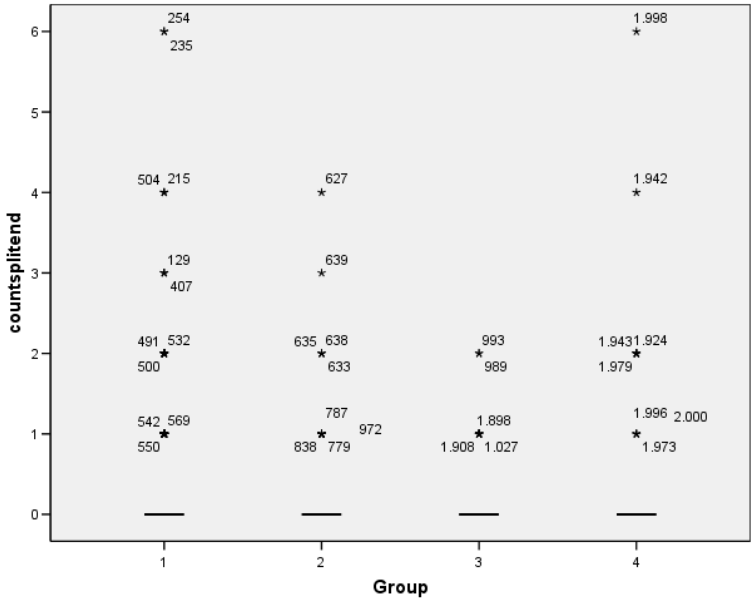


Figure C.45: Box plot for TODO unstructured start and end errors by group

C.3 Box plots filtered by error

This section shows box plots of each variable disaggregated by the variable *hasErrors*. The boolean variables *Error*, *Reduced*, *Interpretable*, and *countProM* are not included since box plots are made for interval scale.

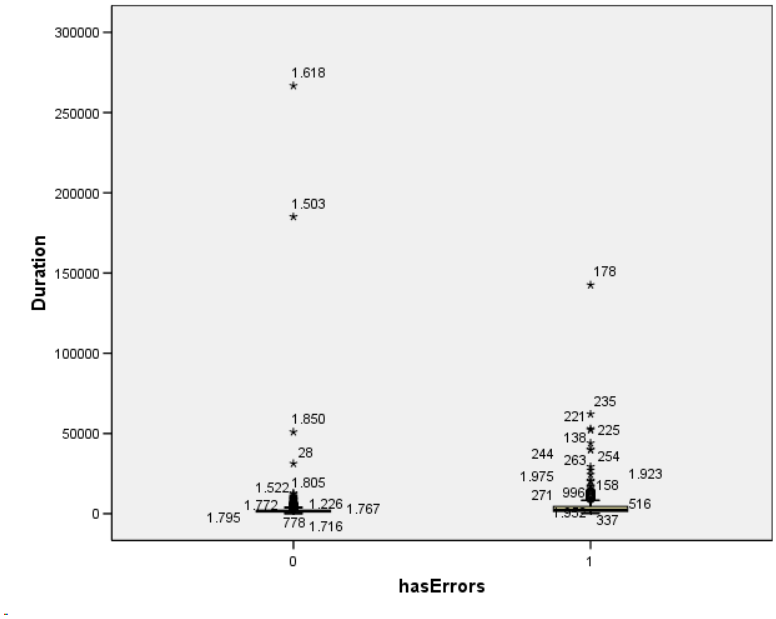


Figure C.46: Box plot for duration by error

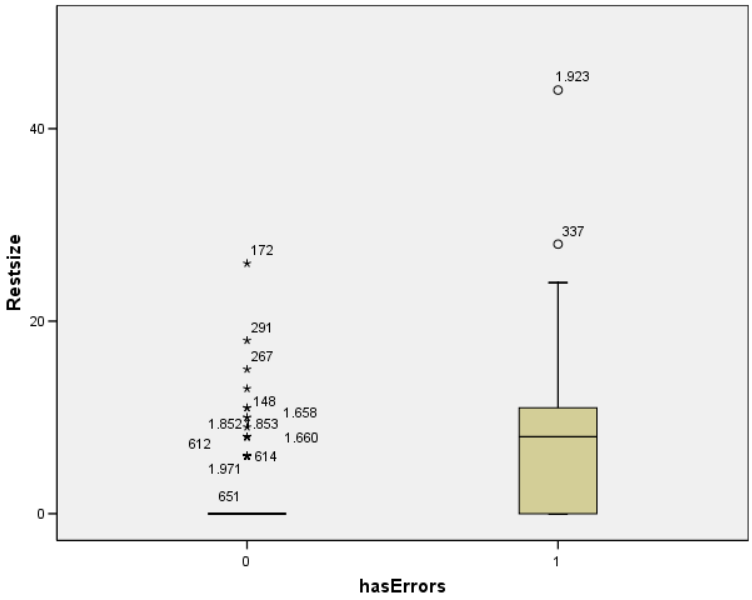


Figure C.47: Box plot for restsize by error

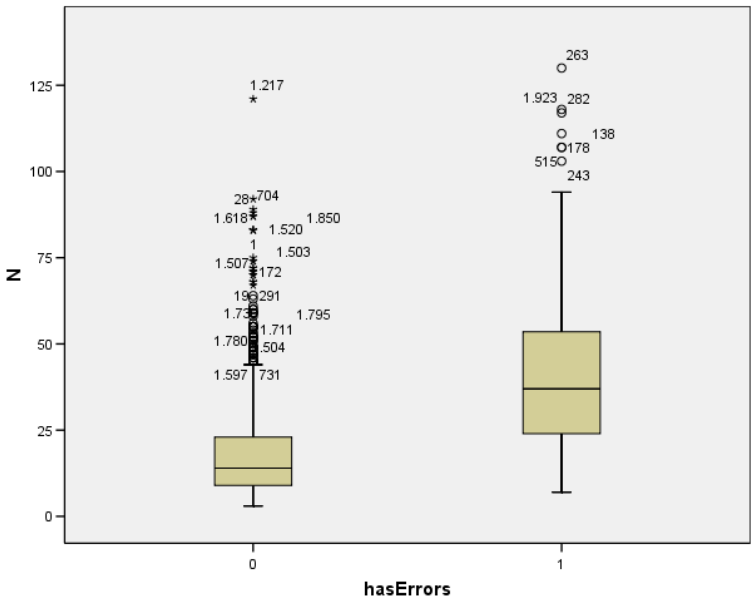


Figure C.48: Box plot for nodes N by error

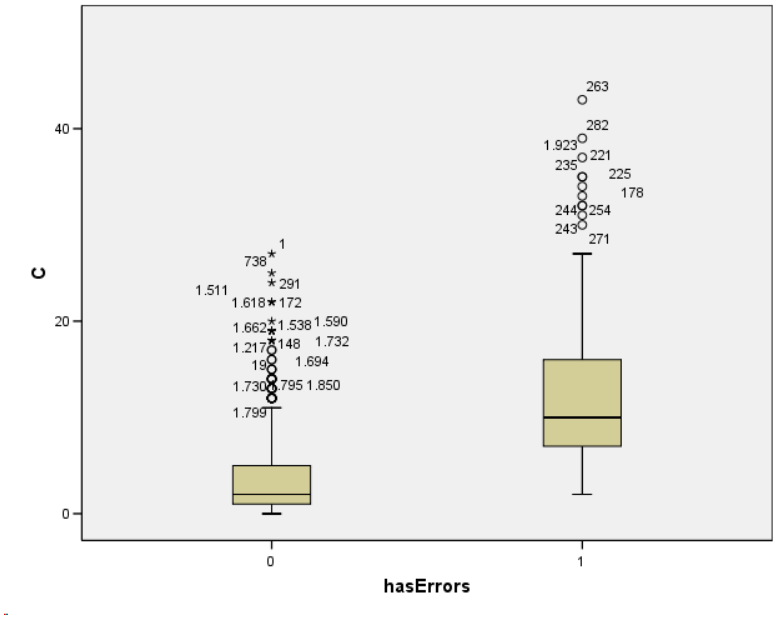


Figure C.49: Box plot for connectors C by error

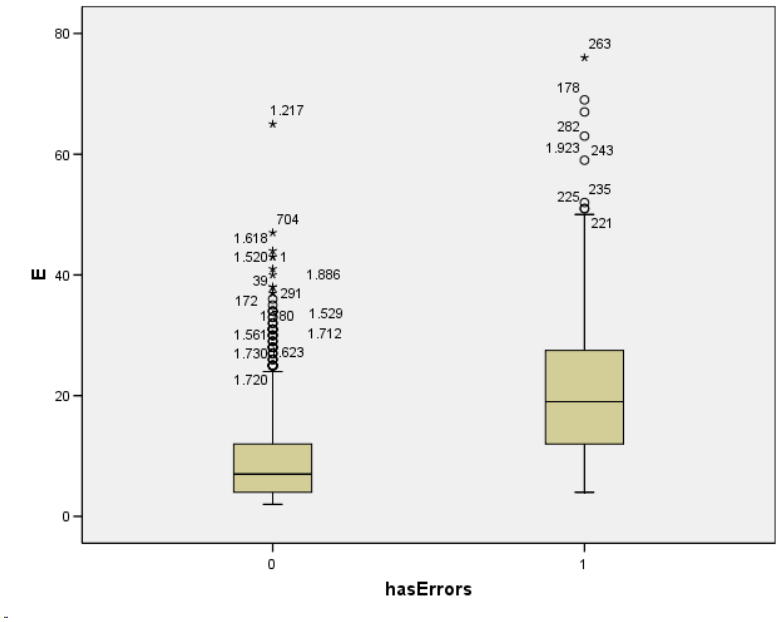


Figure C.50: Box plot for events E by error

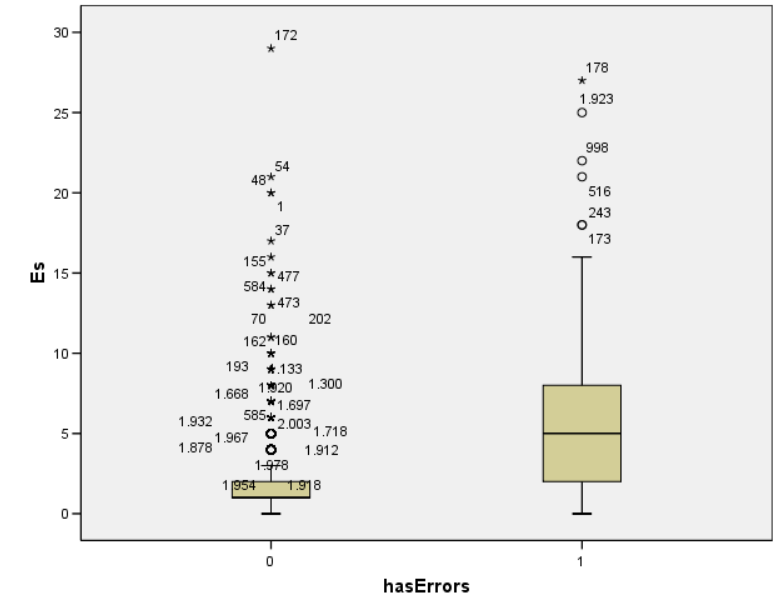


Figure C.51: Box plot for start events Es by error

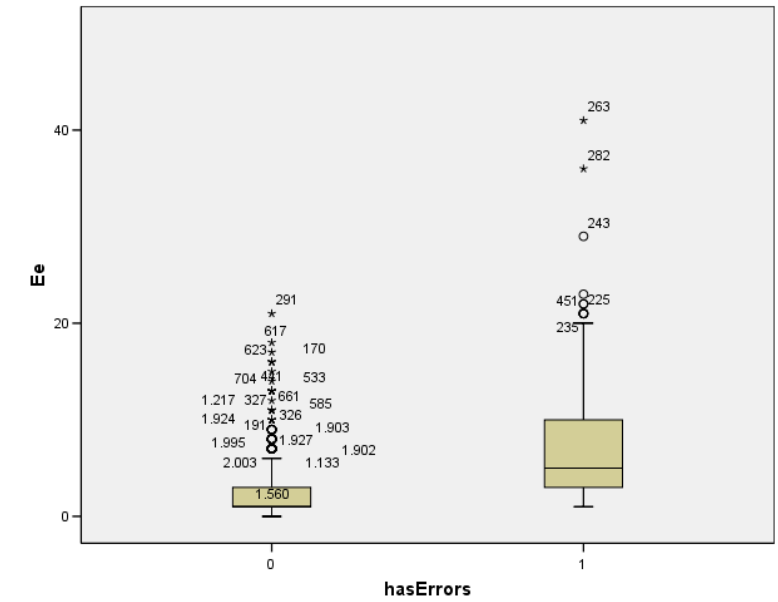


Figure C.52: Box plot for end events Ee by error

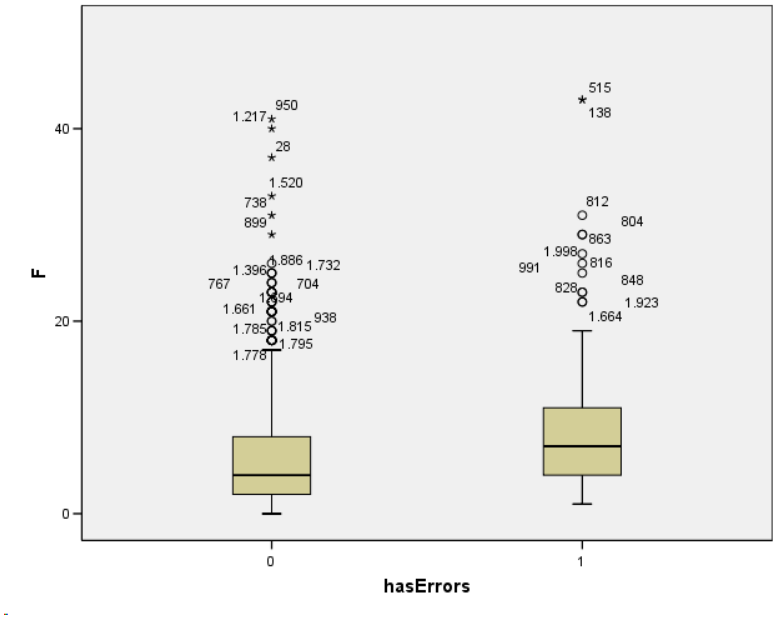


Figure C.53: Box plot for functions F by error

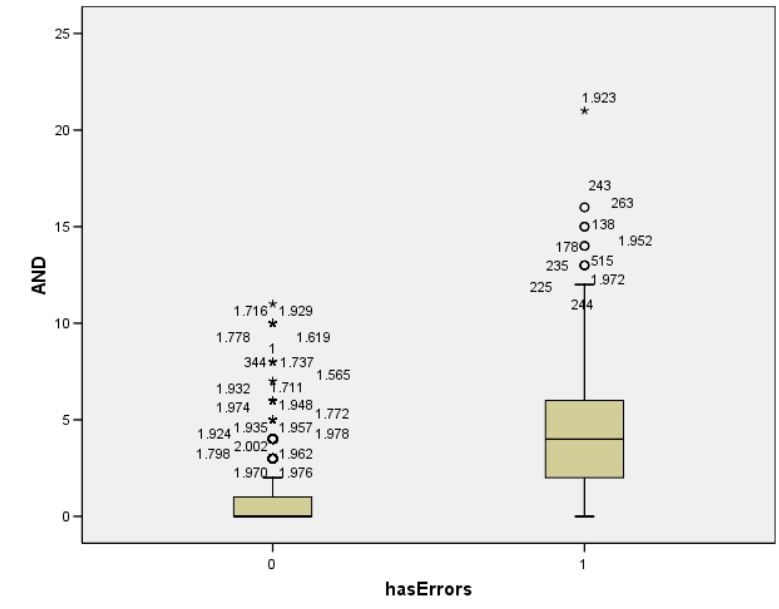


Figure C.54: Box plot for AND-connectors by error

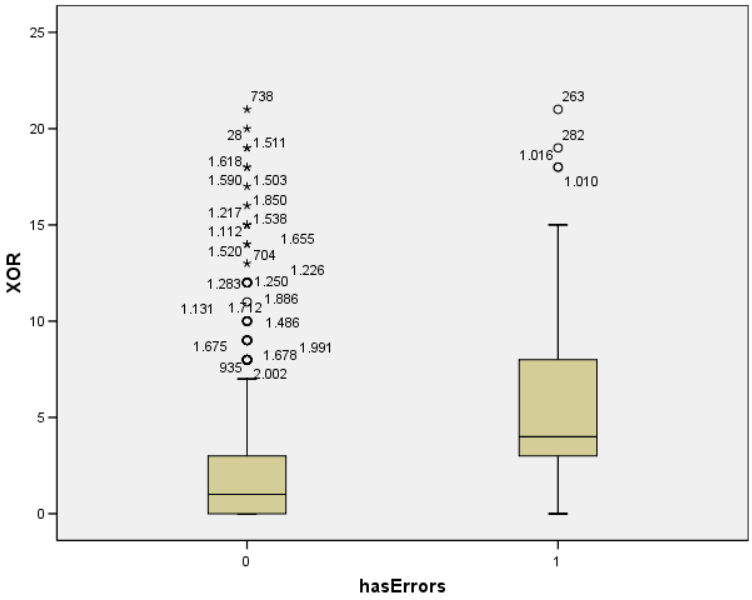


Figure C.55: Box plot for XOR-connectors by error

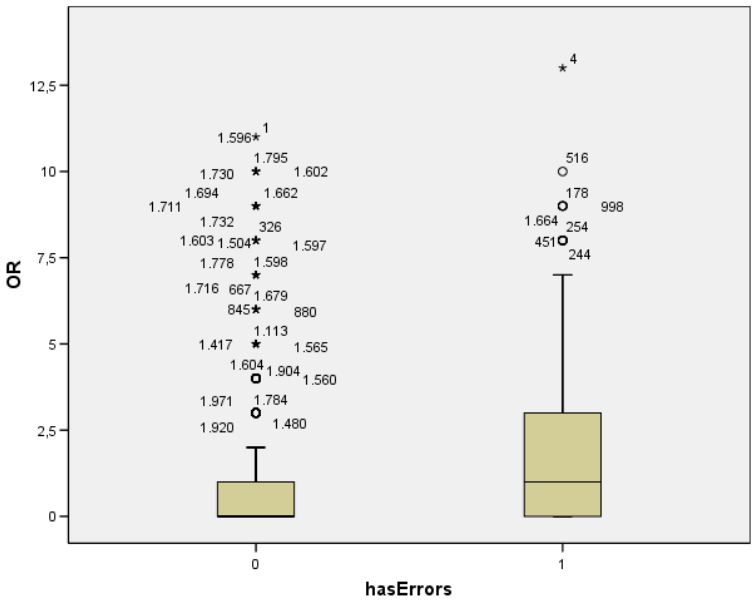


Figure C.56: Box plot for OR-connectors by error

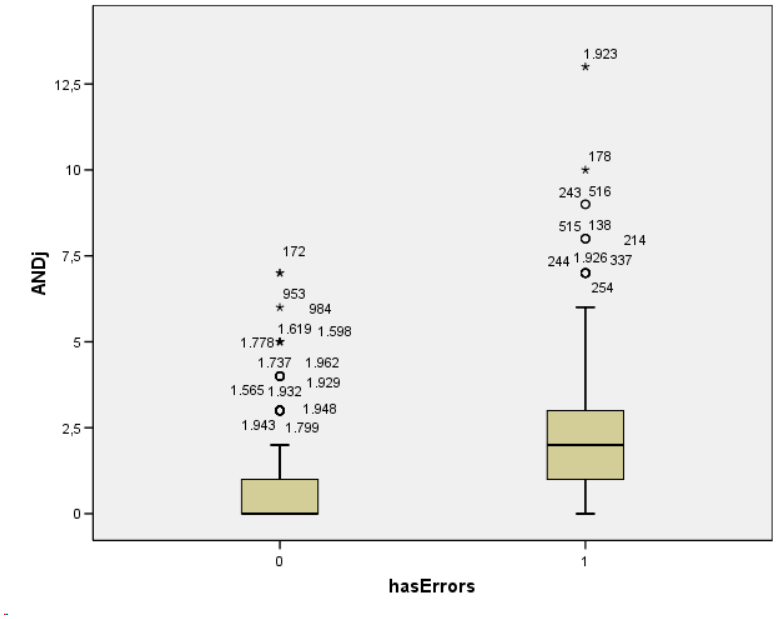


Figure C.57: Box plot for AND-joins by error

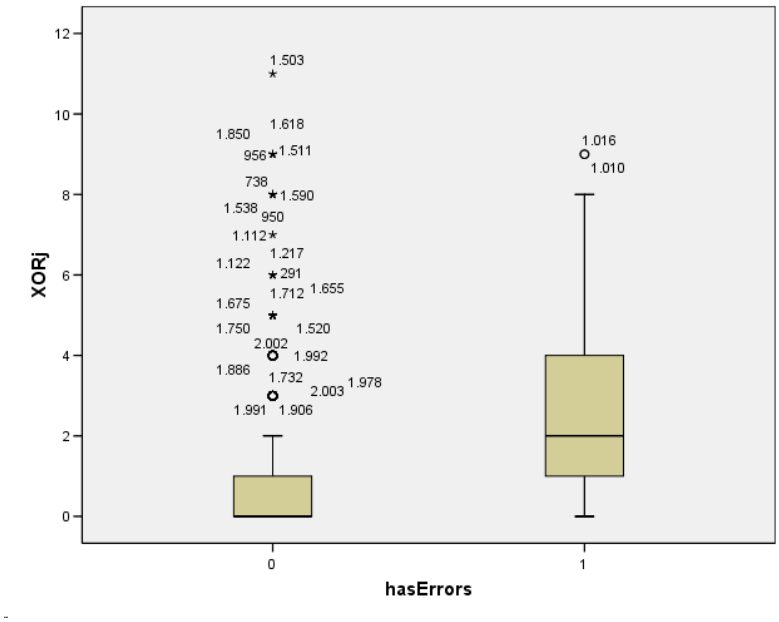


Figure C.58: Box plot for XOR-joins by error

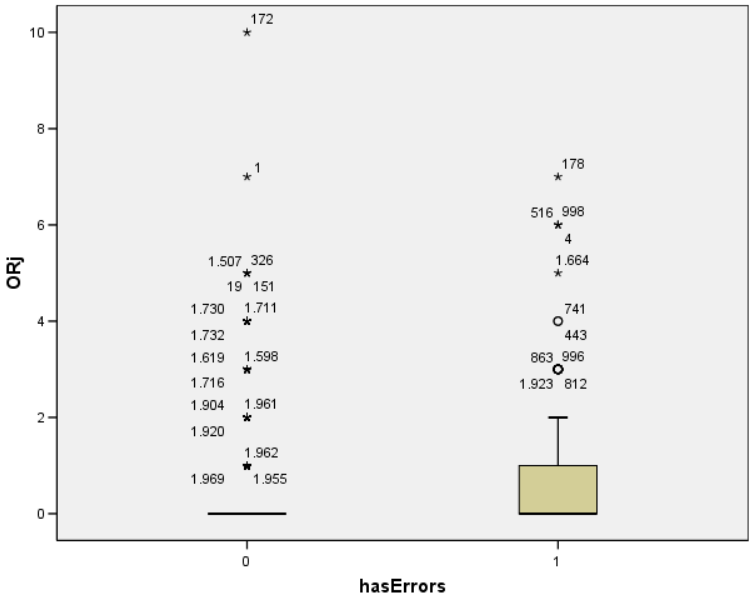


Figure C.59: Box plot for OR-joins by error

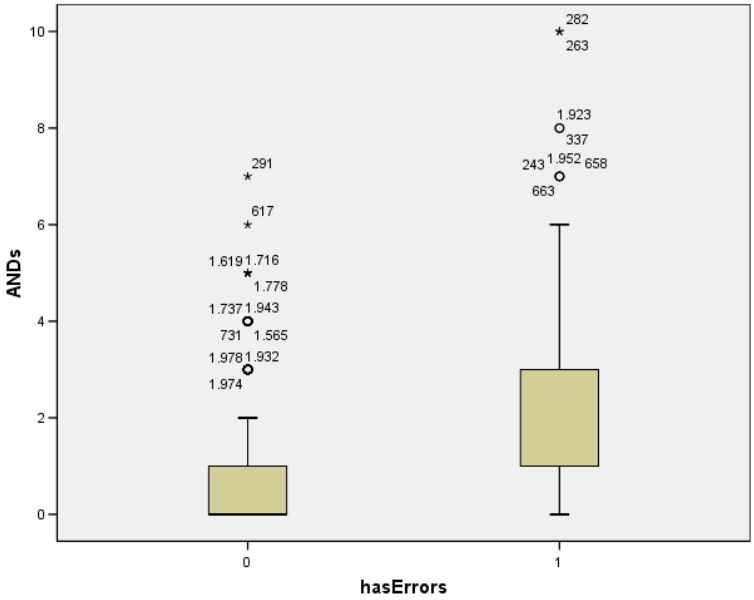


Figure C.60: Box plot for AND-splits by error

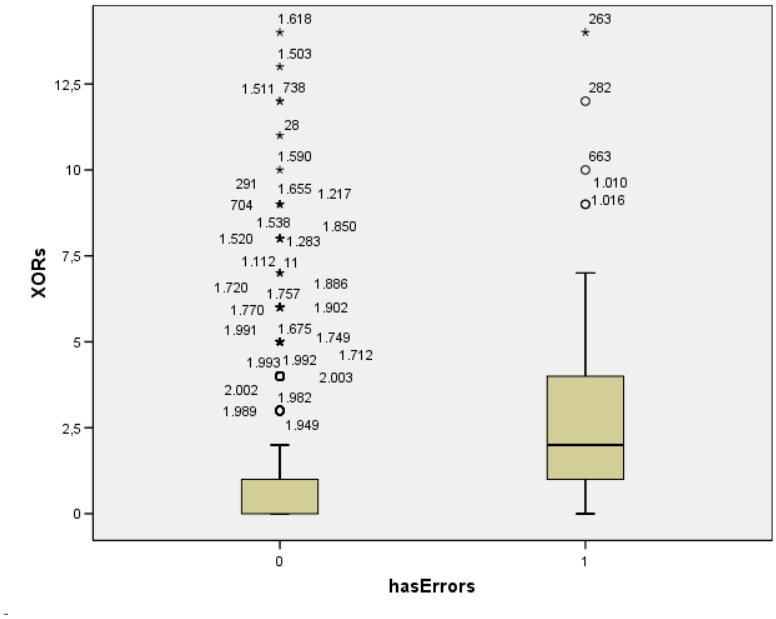


Figure C.61: Box plot for XOR-splits by error

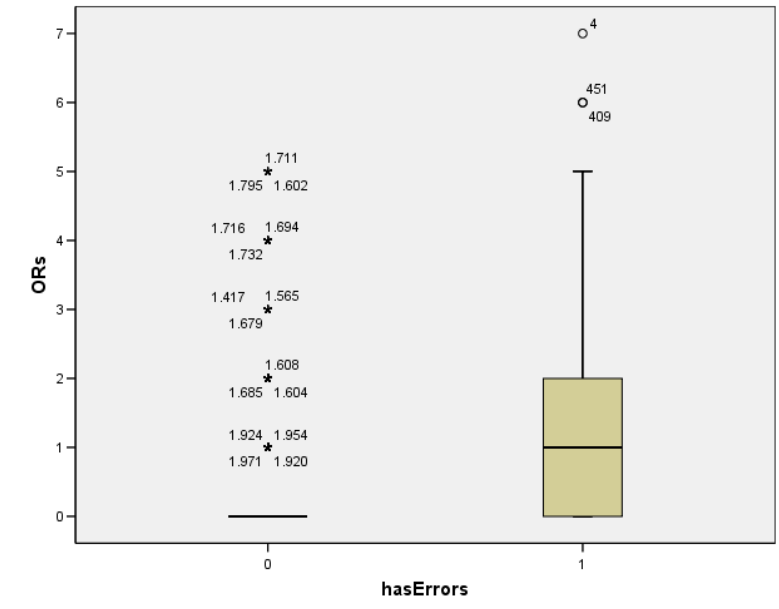


Figure C.62: Box plot for OR-splits by error

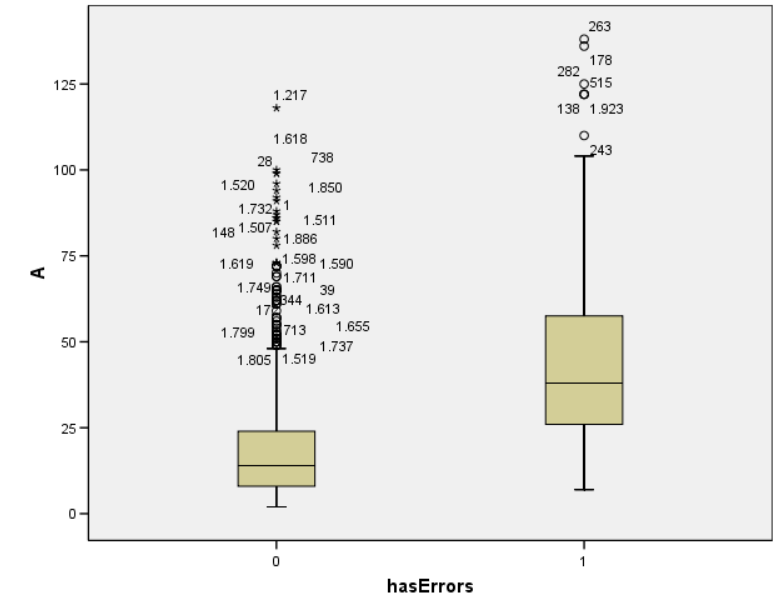


Figure C.63: Box plot for arcs A by error

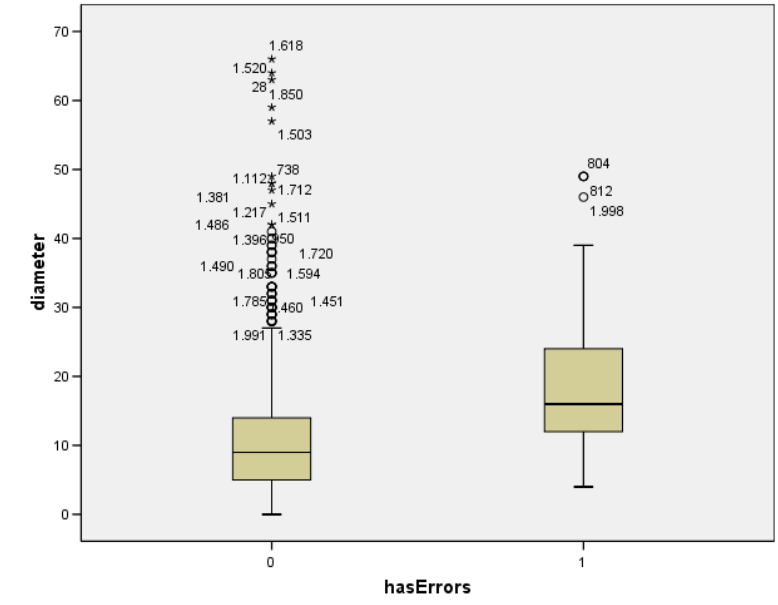


Figure C.64: Box plot for diameter by error

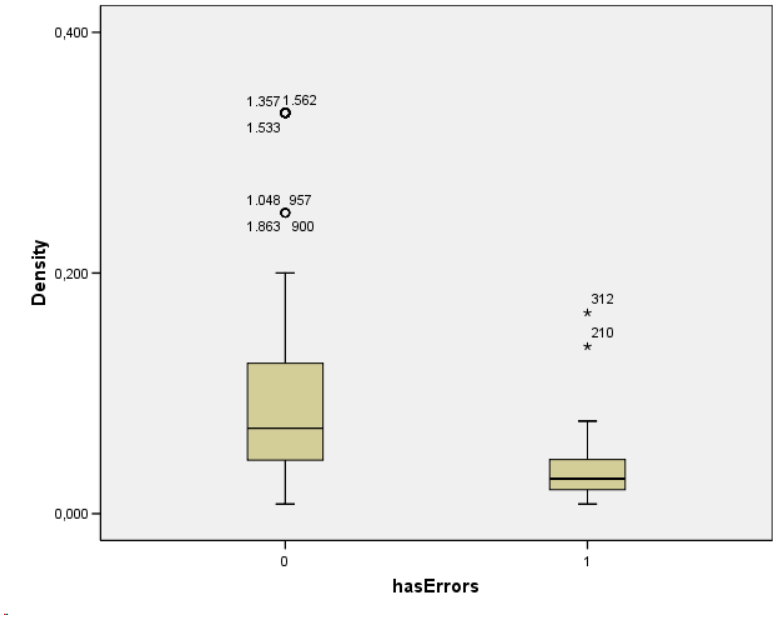


Figure C.65: Box plot for density by error

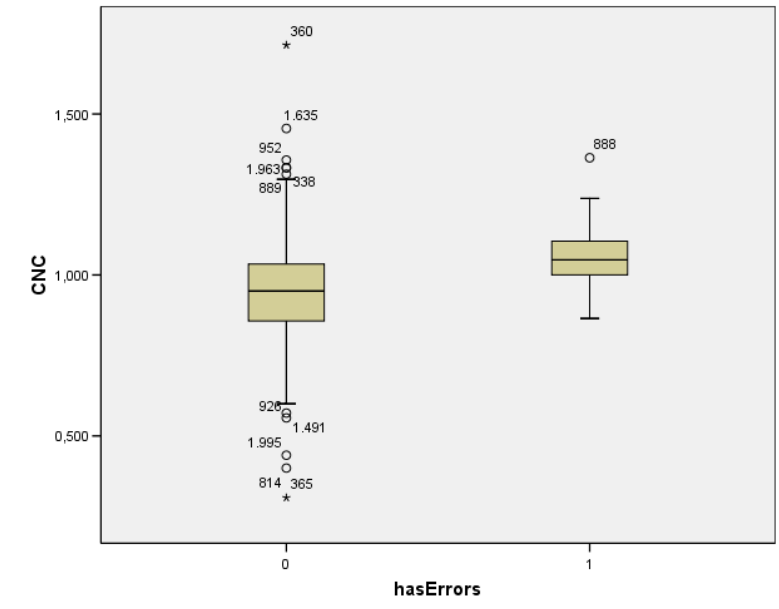


Figure C.66: Box plot for coefficient of connectivity CNC by error

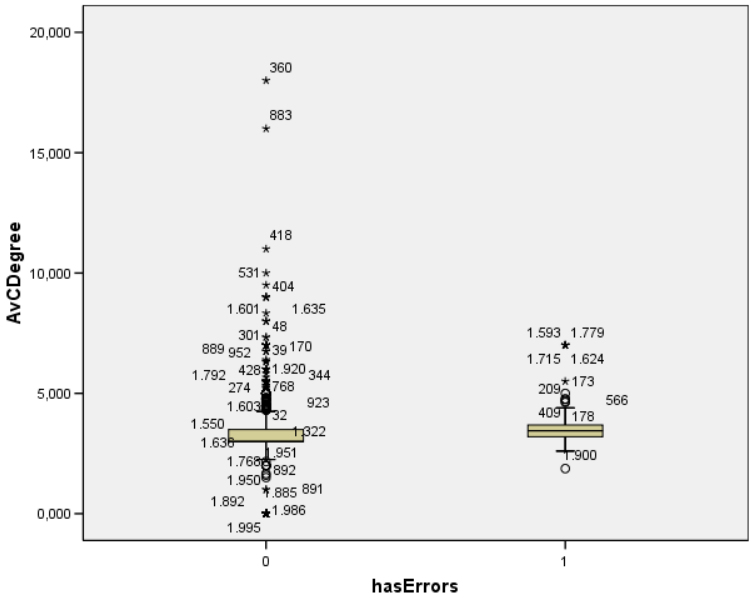


Figure C.67: Box plot for average connector degree by error

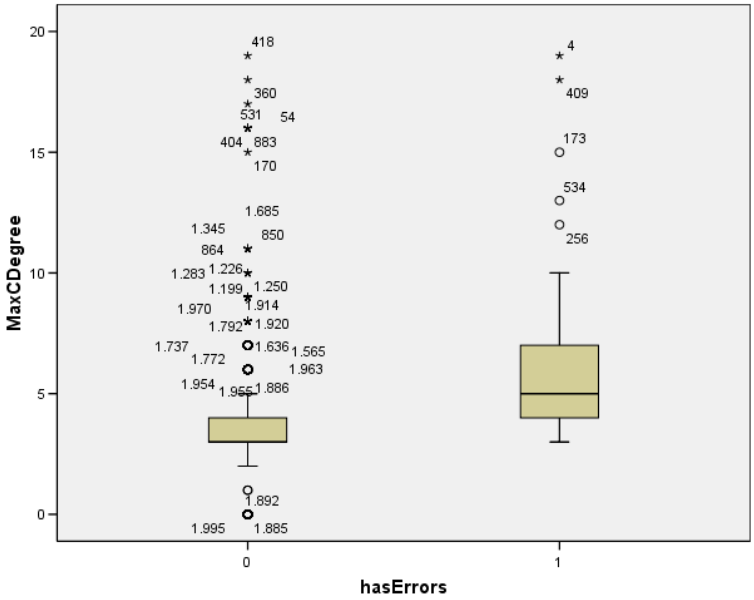


Figure C.68: Box plot for maximum connector degree by error

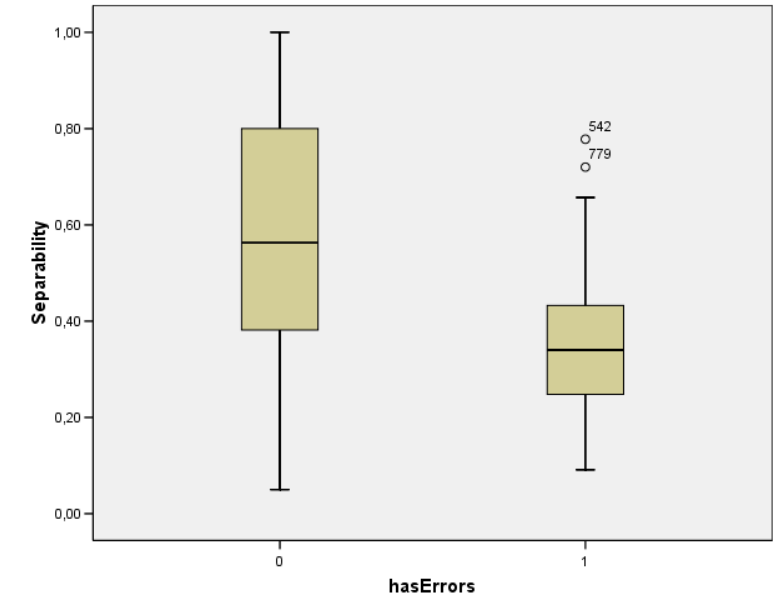


Figure C.69: Box plot for separability by error

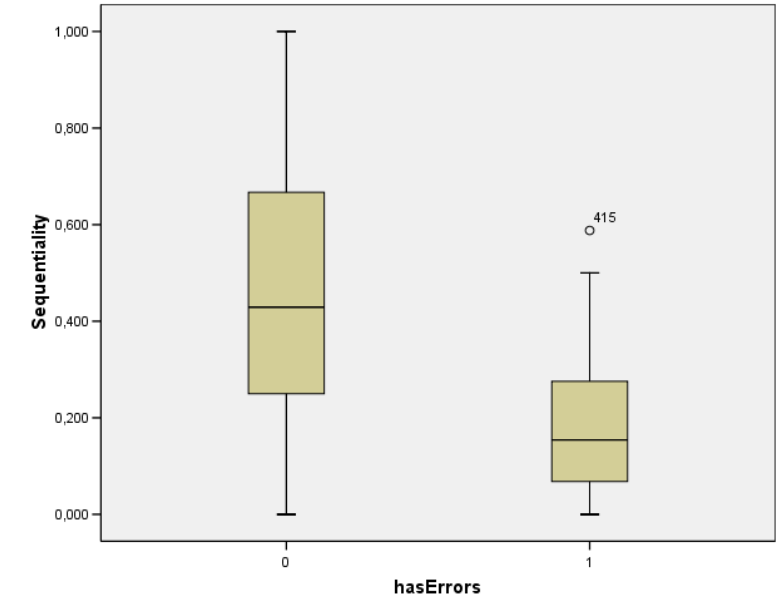


Figure C.70: Box plot for sequentiality by error

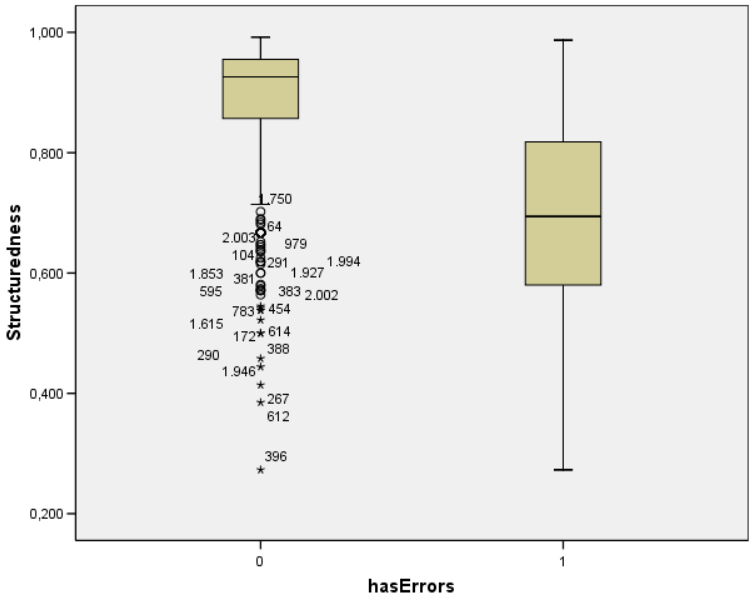


Figure C.71: Box plot for structuredness by error

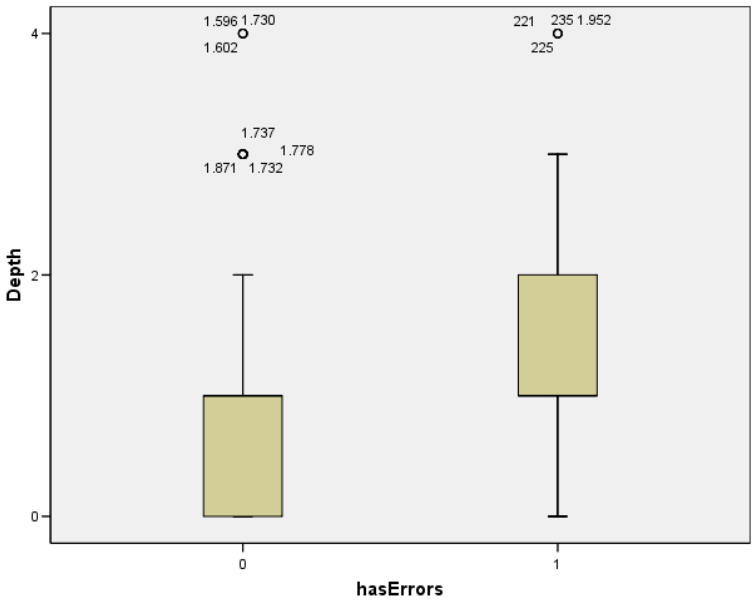


Figure C.72: Box plot for depth by error

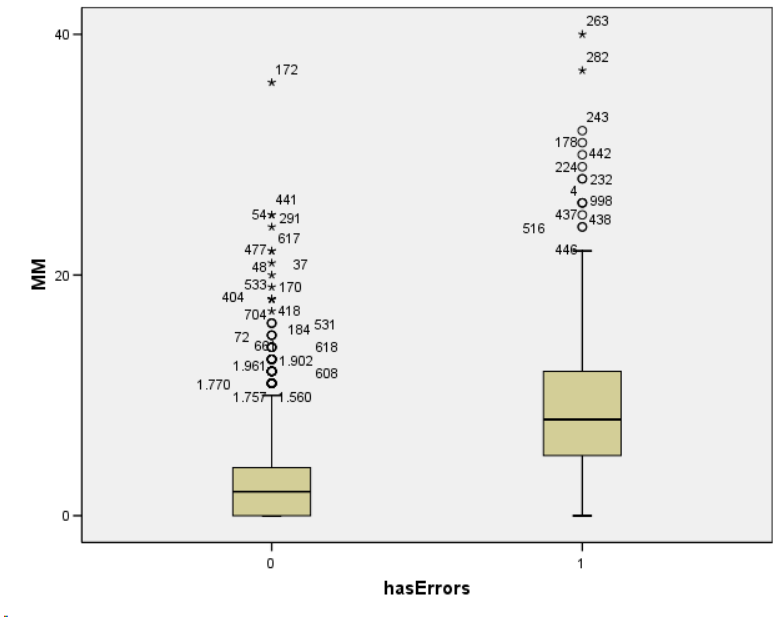


Figure C.73: Box plot for connector mismatch MM by error

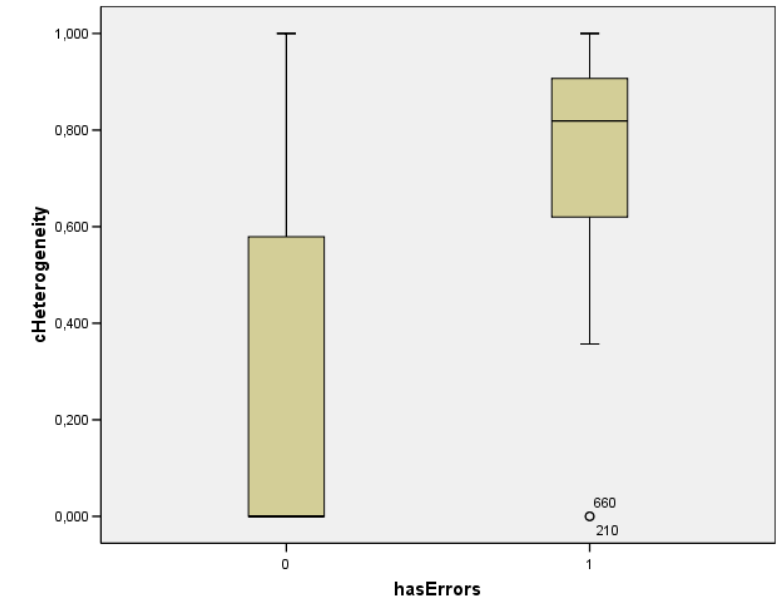


Figure C.74: Box plot for connector heterogeneity by error

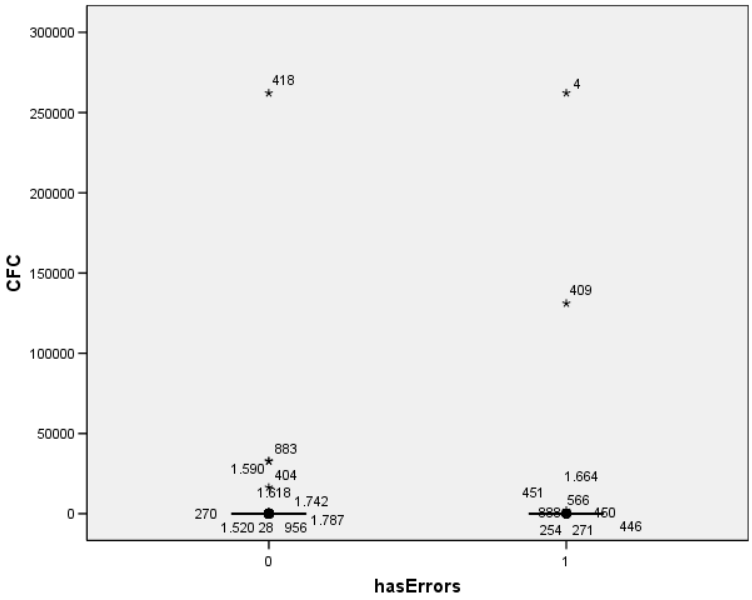


Figure C.75: Box plot for control flow complexity CFC by error

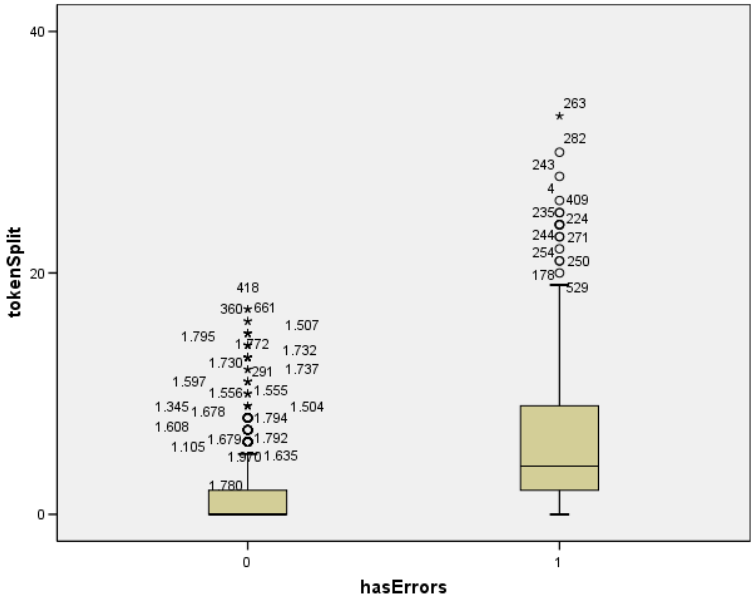


Figure C.76: Box plot for token split by error

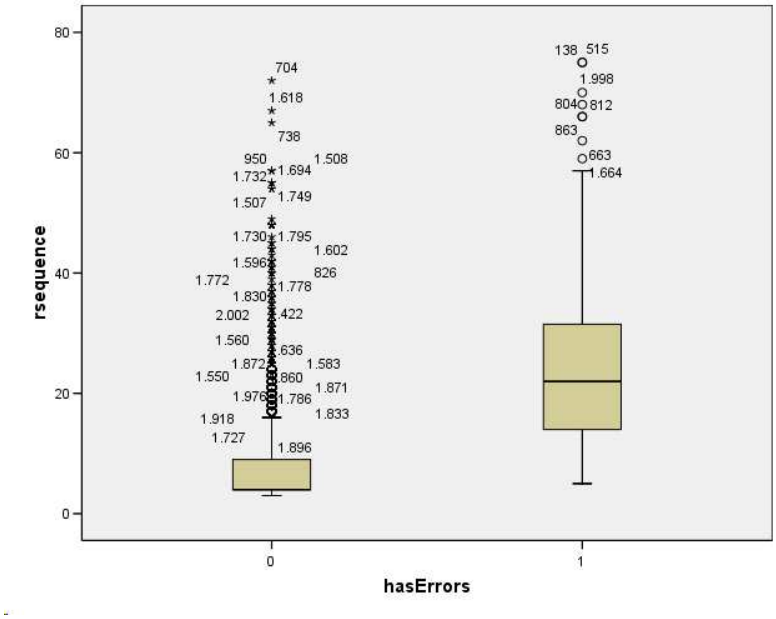


Figure C.77: Box plot for trivial construct rule application by error

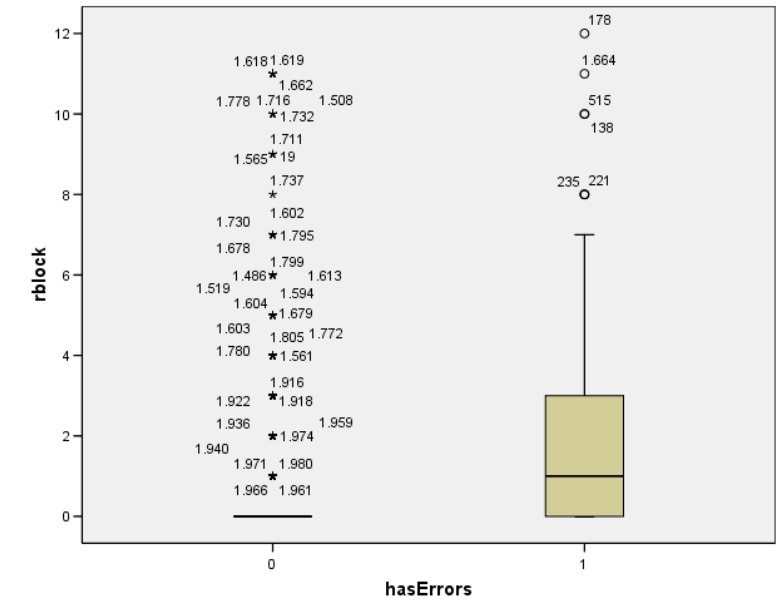


Figure C.78: Box plot for structured block rule application by error

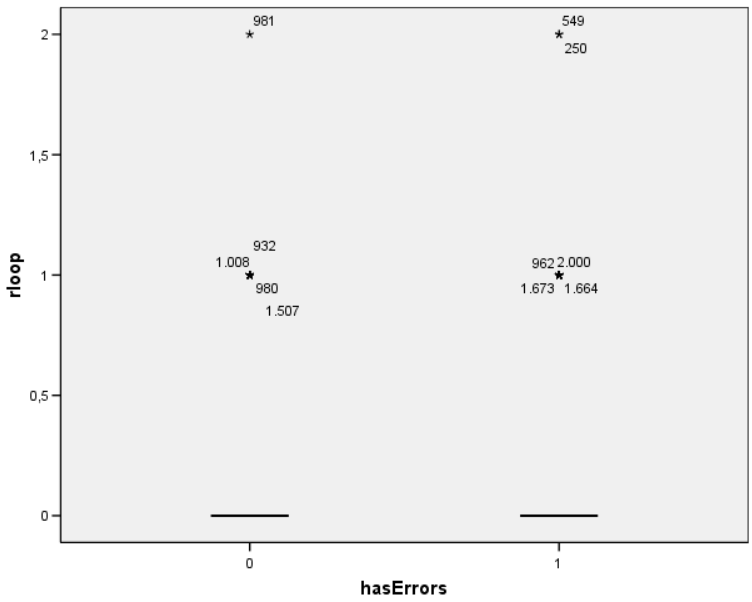


Figure C.79: Box plot for structured loop rule application by error

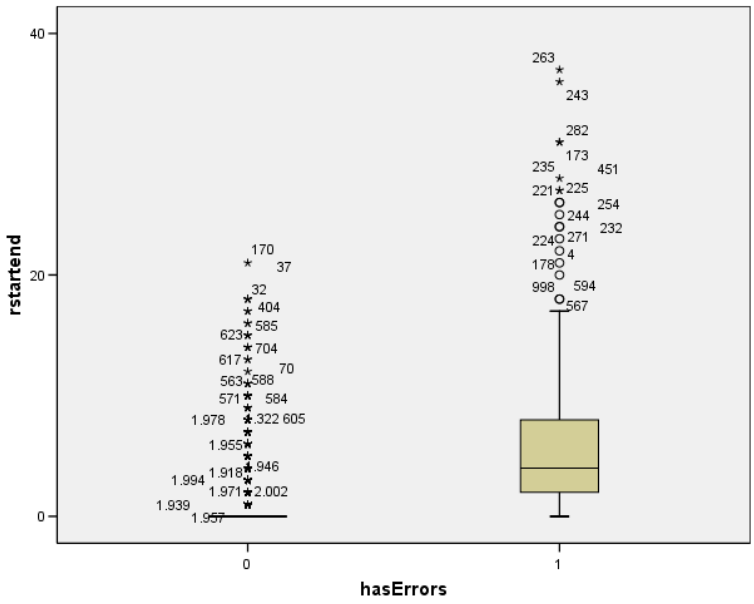


Figure C.80: Box plot for structured start and end rule application by error

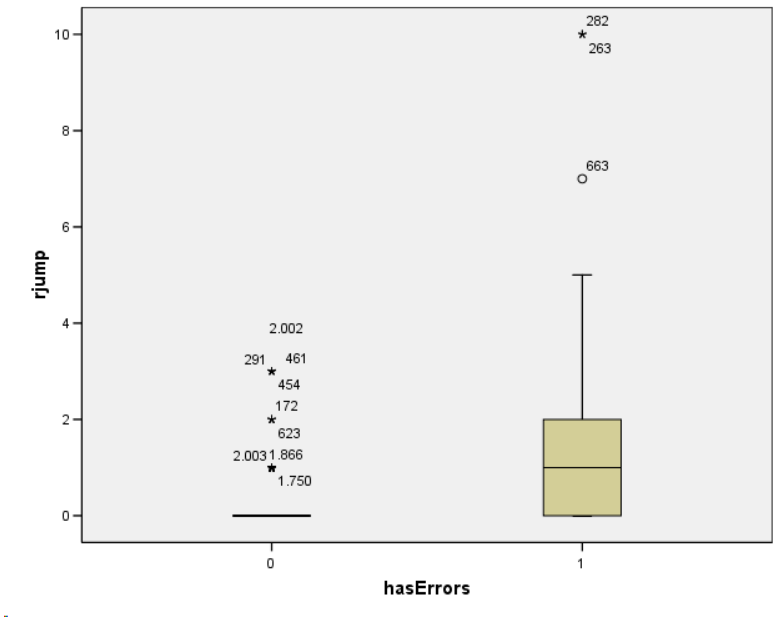


Figure C.81: Box plot for unstructured start and end rule application by error

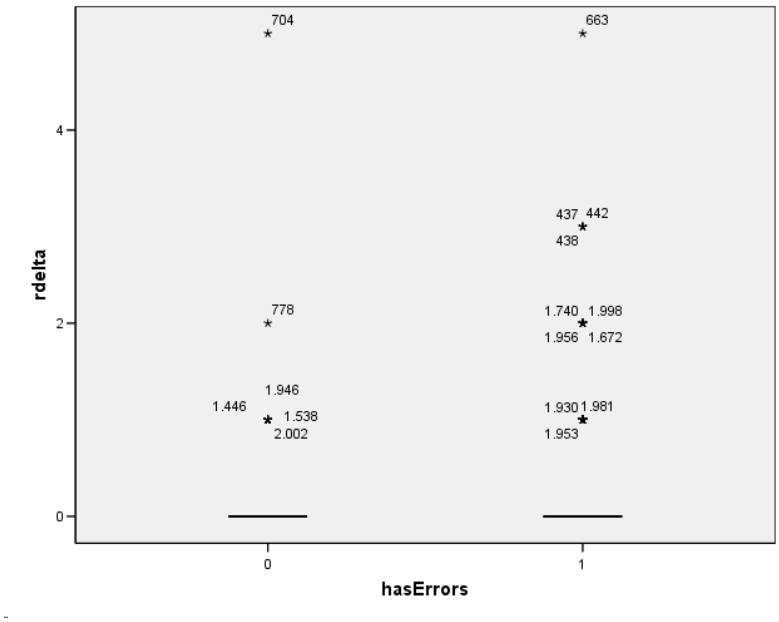


Figure C.82: Box plot for delta rule application by error

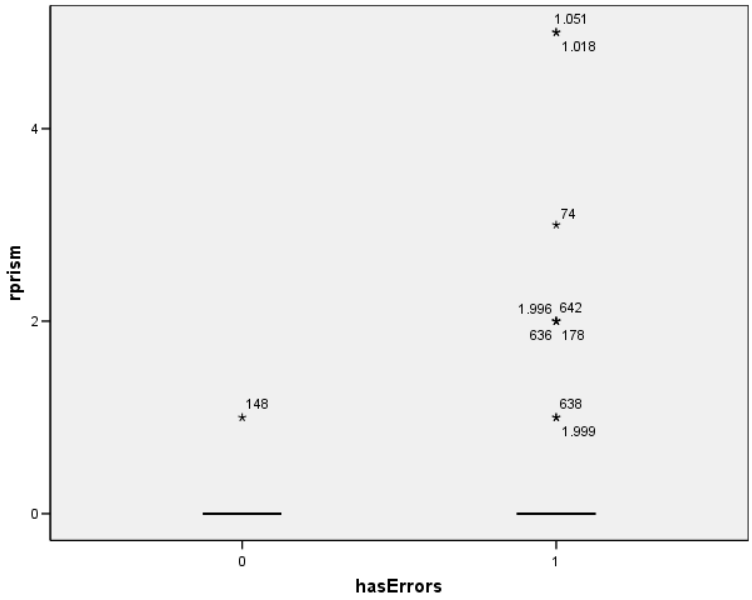


Figure C.83: Box plot for prism rule application by error

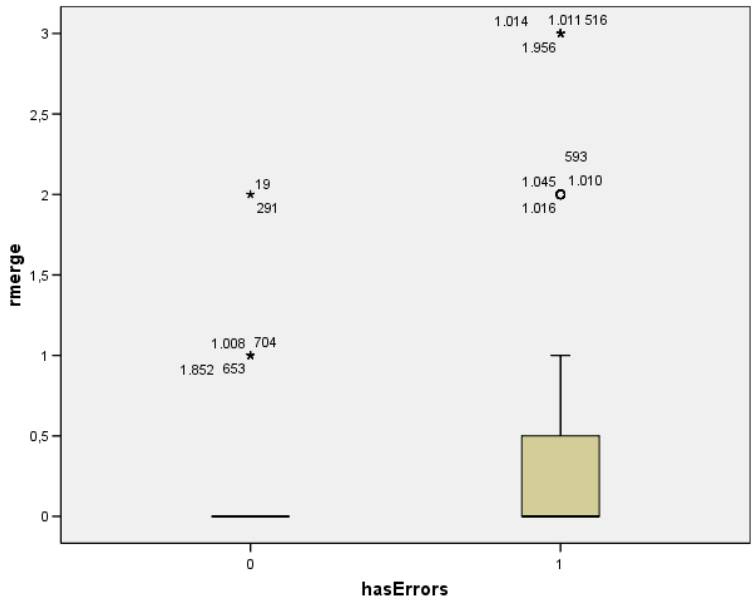


Figure C.84: Box plot for connector merge rule application by error

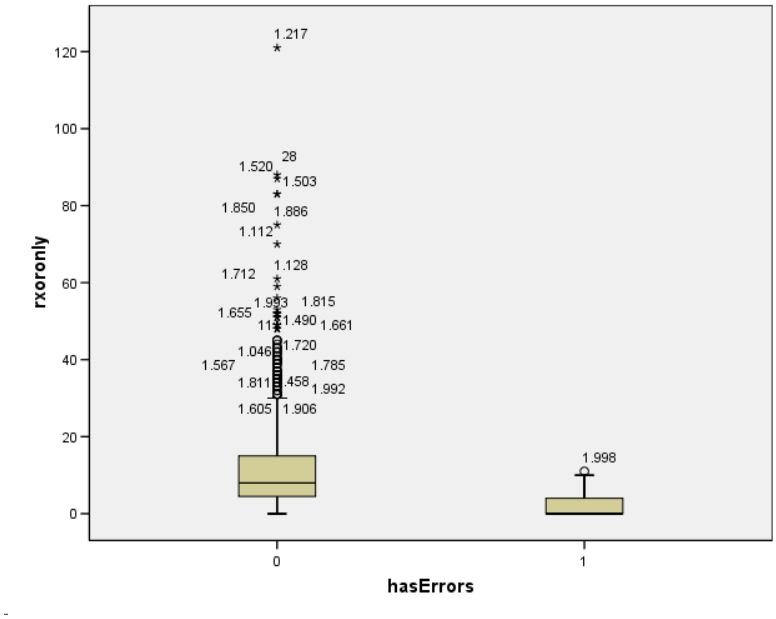


Figure C.85: Box plot for homogeneous rule application by error

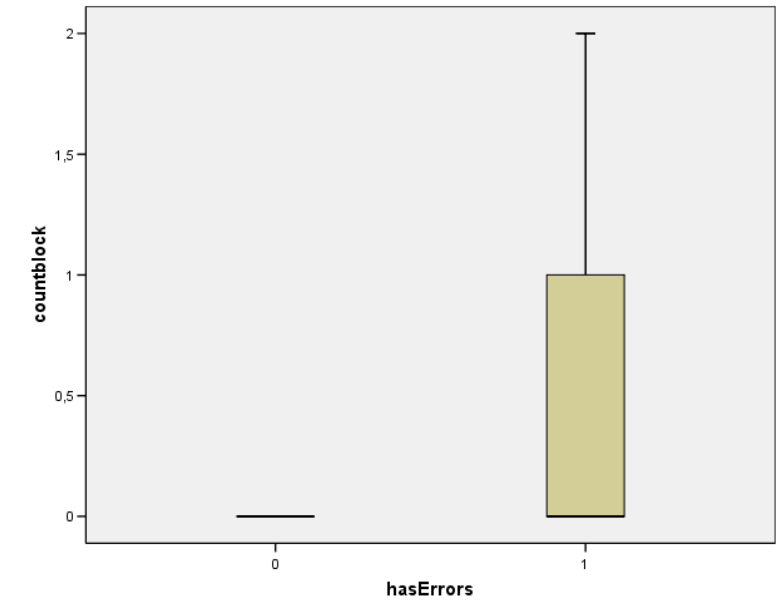


Figure C.86: Box plot for structured block errors by error

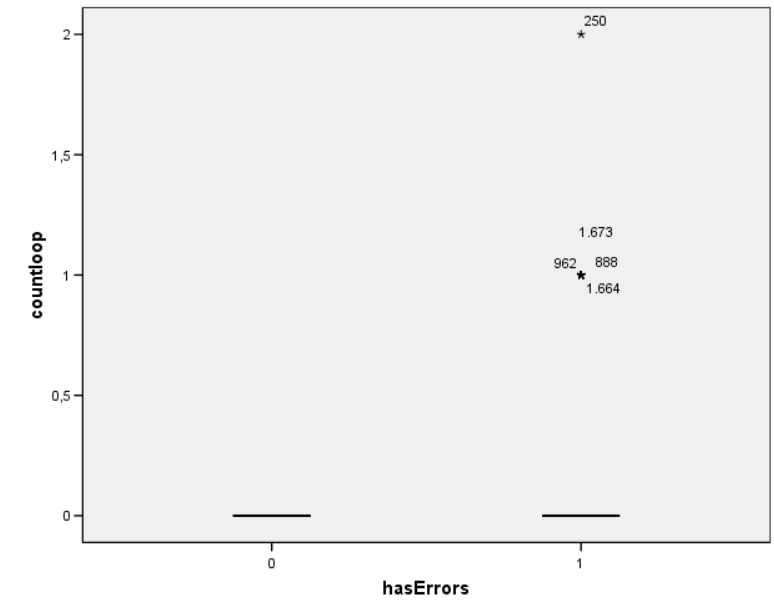


Figure C.87: Box plot for structured loop errors by error

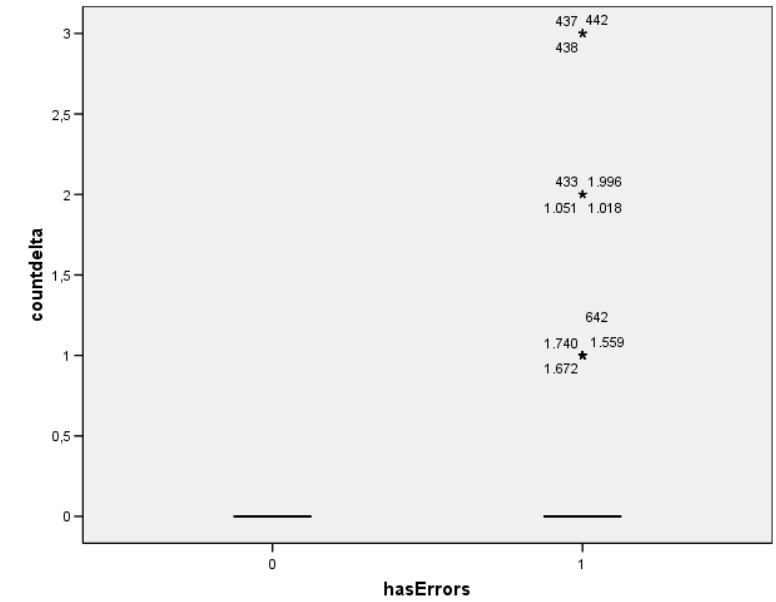


Figure C.88: Box plot for delta errors by error

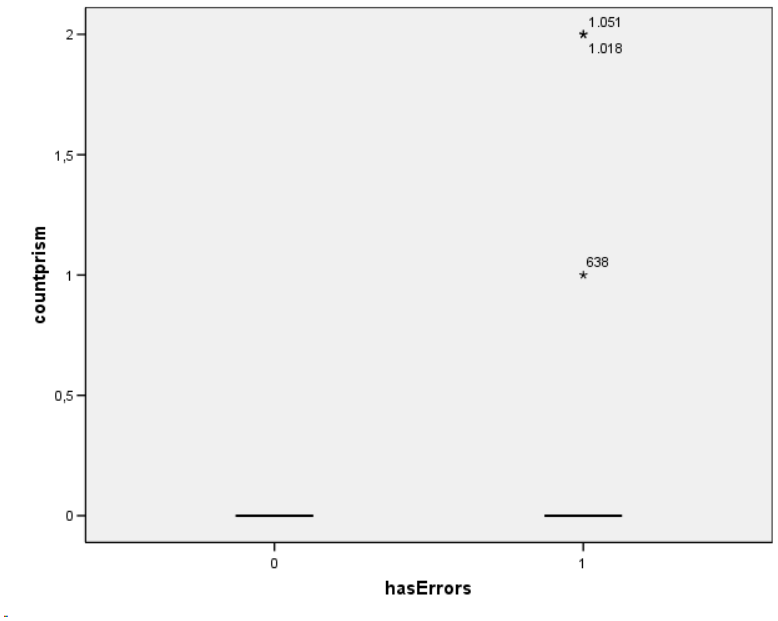


Figure C.89: Box plot for prism errors by error

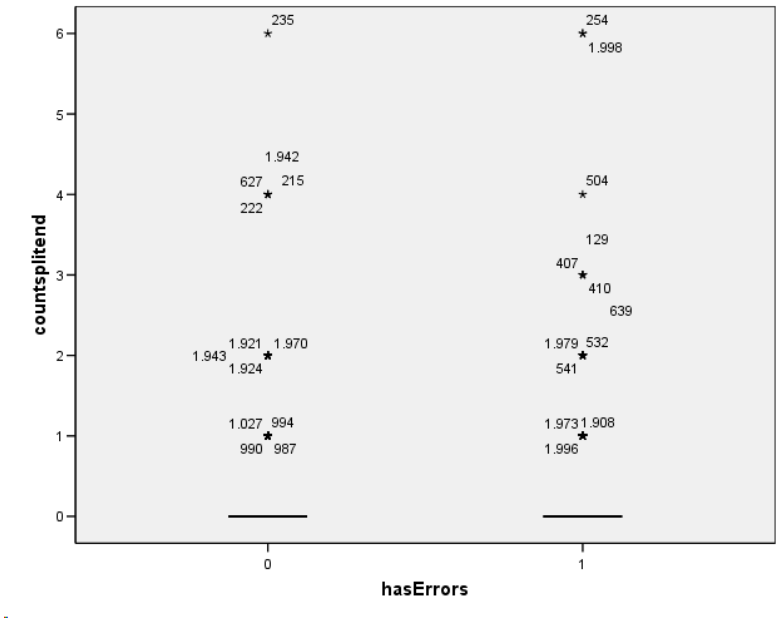


Figure C.90: Box plot for unstructured start and end errors by error

Table C.3: Results of Kolmogorov-Smironov test

	Mean	Std. Dev.	Z	Sig.		Mean	Std. Dev.	Z	Sig.
N	20,71	16,84	6,55	0,00	A	21,11	18,87	6,96	0,00
C	4,27	5,01	9,11	0,00	Sequentiality	0,46	0,31	6,04	0,00
E	10,47	8,66	7,35	0,00	CNC	0,96	0,13	4,91	0,00
Es	2,43	2,70	13,08	0,00	Density	0,09	0,07	7,00	0,00
Ee	2,77	3,20	12,80	0,00	tokenSplit	1,82	3,53	13,57	0,00
F	5,98	4,94	7,29	0,00	AvCDegree	2,88	1,60	14,49	0,00
AND	1,26	2,24	12,81	0,00	MaxCDegree	3,56	2,40	10,34	0,00
XOR	2,25	3,00	10,15	0,00	MM	3,31	4,55	10,45	0,00
OR	0,76	1,54	15,79	0,00	CYC	0,01	0,08	23,59	0,00
ANDj	0,63	1,23	16,28	0,00	Separability	0,56	0,27	4,73	0,00
XORj	1,01	1,46	11,54	0,00	Depth	0,70	0,74	12,05	0,00
ORj	0,37	0,82	18,98	0,00	Structuredness	0,88	0,11	9,01	0,00
ANDs	0,62	1,17	16,14	0,00	CFC	382,62	8849,48	22,11	0,00
XORs	1,24	1,75	11,54	0,00	cHeterogeneity	0,28	0,35	16,66	0,00
ORs	0,37	0,86	19,32	0,00	diameter	11,45	8,21	5,98	0,00

C.4 Analysis of Variance for Metrics grouped by hasErrors

This section summarizes the result of the analysis of variance for metrics grouped by hasErrors. First, we conduct the Kolmogorov-Smirnov test to verify that all variables follow a normal distribution. Then, we summarize the results of the analysis of variance showing that the mean values are significantly different for all metrics.

Table C.4: Analysis of Variance Results ordered by F-Statistic Values

	F	Sig.		F	Sig.
C	884,41	0,00	Depth	286,19	0,00
ANDj	824,72	0,00	ORs	264,28	0,00
AND	819,96	0,00	XORs	232,24	0,00
Structuredness	780,13	0,00	Sequentiality	223,45	0,00
MM	627,43	0,00	MaxCDegree	198,20	0,00
cHeterogeneity	585,51	0,00	diameter	180,17	0,00
E	563,04	0,00	OR	176,35	0,00
Ee	540,36	0,00	Separability	172,92	0,00
N	532,05	0,00	Density	156,89	0,00
A	518,24	0,00	CNC	137,23	0,00
ANDs	502,87	0,00	CYC	124,69	0,00
tokenSplit	471,12	0,00	F	66,59	0,00
Es	424,41	0,00	ORj	64,25	0,00
XORj	344,48	0,00	AvCDegree	44,86	0,00
XOR	331,22	0,00	CFC	6,95	0,01

C.5 Correlation between hasErrors and Metrics

This section shows the correlation between hasErrors and the different metrics, first as Pearson's correlation coefficient (Table C.5) and afterwards as Spearman's rank correlation coefficient (Table C.6).

Table C.5: Pearson Correlation between hasErrors and Metrics (below significance)

	hasErrors		hasErrors
Duration	0,13	ORs	0,34
	0,00		0,00
Restsize	0,62	A	0,45
	0,00		0,00
N	0,46	diameter	0,29
	0,00		0,00
C	0,55	Density	-0,27
	0,00		0,00
E	0,47	CNC	0,25
	0,00		0,00
Es	0,42	AvCDegree	0,15
	0,00		0,00
Ee	0,46	MaxCDegree	0,30
	0,00		0,00
F	0,18	Separability	-0,28
	0,00		0,00
AND	0,54	Sequentiality	-0,32
	0,00		0,00
XOR	0,38	Structuredness	-0,53
	0,00		0,00
OR	0,28	Depth	0,35
	0,00		0,00
ANDj	0,54	MM	0,49
	0,00		0,00
XORj	0,38	cHeterogeneity	0,48
	0,00		0,00
ORj	0,18	CFC	0,06
	0,00		0,01
ANDs	0,45	CYC	0,24
	0,00		0,00
XORs	0,32	tokenSplit	0,44
	0,00		0,00

Table C.6: Spearman Rank Correlation between hasErrors and Metrics (below significance)

	hasErrors		hasErrors
Duration	0,19	ORs	0,31
	0,00		0,00
Restsize	0,66	A	0,38
	0,00		0,00
N	0,38	diameter	0,30
	0,00		0,00
C	0,43	Density	-0,37
	0,00		0,00
E	0,38	CNC	0,28
	0,00		0,00
Es	0,35	AvCDegree	0,23
	0,00		0,00
Ee	0,38	MaxCDegree	0,33
	0,00		0,00
F	0,19	Separability	-0,29
	0,00		0,00
AND	0,45	Sequentiality	-0,35
	0,00		0,00
XOR	0,35	Structuredness	-0,36
	0,00		0,00
OR	0,30	Depth	0,34
	0,00		0,00
ANDj	0,48	MM	0,42
	0,00		0,00
XORj	0,33	cHeterogeneity	0,46
	0,00		0,00
ORj	0,15	CFC	0,39
	0,00		0,00
ANDs	0,37	CYC	0,30
	0,00		0,00
XORs	0,31	tokenSplit	0,38
	0,00		0,00

Appendix D

Logistic Regression Results

This appendix gathers details of the logistic regression analysis. In particular, Section D.1 gives a tabular overview of the collinearity analysis of the variables. This analysis led to a reduction of the variable set in such a way that S_N is the only remaining count metric for size. Section C.2 presents the results of univariate logistic regression models of all variables of the reduced set. These univariate models show that there is no constant in a multivariate model required since the constant is not significantly different from zero in two models (see Wald statistic). Furthermore, the control flow complexity is not significantly different from zero in both models with and without constant. Therefore, it is dropped from the variables list. Section D.3 shows results from the multivariate logistic regression analysis.

D.1 Collinearity Analysis

This section gives the results of the collinearity analysis. The absence of collinearity is not a hard criterion for the applicability of logistic regression, but it is desirable. In a

Table D.1: Tolerance Values for Metrics

	Tolerance		Tolerance
N	0.0000	A	0.0017
C	0.0000	diameter	0.1217
E	0.0062	Density	0.1978
Es	0.1269	CNC	0.1362
Ee	0.0607	AvCDegree	0.1151
F	0.0228	MaxCDegree	0.0792
AND	0.0064	Separability	0.2539
XOR	0.0123	Sequentiality	0.1377
OR	0.0125	Structuredness	0.5555
ANDj	0.0202	Depth	0.2228
XORj	0.0431	MM	0.2365
ORj	0.0404	cHeterogeneity	0.3824
ANDs	0.0209	CFC	0.6966
XORs	0.0287	CYC	0.8913
ORs	0.0349	tokenSplit	0.0488

Table D.2: Tolerance Values after reducing the Metrics Set

	Tolerance		Tolerance
N	0.0931	Structuredness	0.6225
diameter	0.1564	Depth	0.2606
CNC	0.2570	MM	0.3261
Density	0.2875	cHeterogeneity	0.4241
AvCDegree	0.1283	CFC	0.8073
MaxCDegree	0.1080	CYC	0.9326
Separability	0.2828	tokenSplit	0.3008
Sequentiality	0.2576		

variable set without collinearity every variable should have a tolerance value higher than 0.1, otherwise there is a collinearity problem. In the original variable set (Table D.1) there are several collinearity problems. We dropped the count metrics apart from S_N since they were highly correlated. This resulted in a reduced variable set with almost no collinearity problems (Table D.2). The S_N metric is close to the 0.1 threshold and therefore kept in the metrics set.

D.2 Univariate Logistic Regression

This section presents the results of the univariate logistic regression analysis. In particular we calculated univariate models with and without a constant (see Tables D.3 and D.4). As a conclusion from these models we drop the constant and the control flow complexity CFC for the multivariate analysis. First, the constant is not significantly different from zero (see Wald statistic) in the separability and the sequentiality model which suggests that it is not necessary. Second, the CFC metric is not significantly different from zero (see Wald statistic) in both models with and without constant.

D.3 Multivariate Logistic Regression

Based on a reduced set of variables without CFC we calculated multivariate logistic regression models. Figure D.1 shows that the Hosmer & Lemeshow Test indicates a good fit based on the difference between observed and predicted frequencies. This test should yield a value greater than 5% and this condition is fulfilled by all models from step 3 on. Figure D.2 summarizes the value of Nagelkerke's R^2, a statistic ranging from 0 to 1 that serves as a coefficient of determination. It indicates which fraction of the variability is explained. The figure shows that from step 3 on the value approaches 0.90 which is an excellent value. Figure D.3 and D.4 give the classification tables and the equations of the models in the different steps.

Table D.3: Univariate logistic regression models without constant

	B	Exp(B)	Wald	Hosmer & L.	Nagelkerke R^2
N	-0.440	0.957	0.000	0.000	0.256
diameter	-0.112	0.894	0.000	0.000	0.387
CNC	-2.082	0.013	0.000	0.000	0.637
Density	-41.081	0.000	0.000	0.000	0.771
AvCDegree	-0.532	0.588	0.000	0.000	0.506
MaxCDegree	-0.351	0.704	0.000	0.000	0.396
Separability	-4.657	0.009	0.000	0.000	0.733
Sequentiality	-7.038	0.001	0.000	0.123	0.760
Structuredness	-2.688	0.068	0.000	0.000	0.728
Depth	-0.908	0.403	0.000	0.000	0.193
MM	-0.090	0.914	0.000	0.000	0.066
cHeterogeneity	-1.223	0.294	0.000	0.000	0.085
CFC	0.000	1.000	0.531	0.000	0.000
CYC	0.301	1.352	0.588	0.999	0.000
tokenSplit	-0.067	0.935	0.000	0.000	0.020

Table D.4: Univariate logistic regression models with constant

	Cons.	Exp(Cons.)	Wald	B	Exp(B)	Wald	H. & L.	N. R^2
N	-3.954	0.019	0.000	0.068	1.070	0.000	0.000	0.295
diameter	-3.306	0.037	0.000	0.087	1.091	0.000	0.000	0.132
CNC	-9.411	0.000	0.000	7.294	1472.146	0.000	0.000	0.138
Density	0.634	1.885	0.001	-54.440	0.000	0.000	0.000	0.311
AvCDegree	-3.029	0.048	0.000	0.291	1.338	0.000	0.000	0.042
MaxCDegree	3.575	0.028	0.000	0.344	1.411	0.000	0.000	0.145
Separability	0.027	1.028	0.872	-4.716	0.009	0.000	0.000	0.184
Sequentiality	-0.204	0.815	0.117	-6.391	0.002	0.000	0.262	0.268
Structuredness	7.064	1169.081	0.000	-11.210	0.000	0.000	0.000	0.377
Depth	-3.419	0.033	0.000	1.343	3.830	0.000	0.000	0.208
MM	-3.459	0.031	0.000	0.270	1.310	0.000	0.000	0.318
cHeterogeneity	-4.811	0.008	0.000	5.259	192.361	0.000	0.000	0.413
CFC	-2.115	0.121	0.000	0.000	1.000	0.382	0.000	0.001
CYC	-2.244	0.106	0.000	5.104	164.740	0.000	0.999	0.065
tokenSplit	-2.871	0.057	0.000	0.269	1.308	0.000	0.000	0.235

Hosmer and Lemeshow Test

Step	Chi-square	df	Sig.
1	330,522	8	,000
2	26,819	8	,001
3	4,278	8	,831
4	4,341	8	,825
5	8,101	8	,424
6	9,961	8	,268
7	7,184	8	,517
8	10,573	8	,227
9	7,890	8	,444

Figure D.1: Hosmer and Lemeshow test for multivariate logistic regression

Model Summary

Step	-2 Log likelihood	Cox & Snell R Square	Nagelkerke R Square
1	1178,396[a]	,546	,728
2	768,884[b]	,631	,841
3	584,495[c]	,664	,885
4	554,211[d]	,669	,892
5	528,702[c]	,673	,898
6	521,807[c]	,674	,899
7	515,520[d]	,675	,901
8	511,687[d]	,676	,901
9	513,645[d]	,676	,901

a. Estimation terminated at iteration number 5 because parameter estimates changed by less than ,001.

b. Estimation terminated at iteration number 6 because parameter estimates changed by less than ,001.

c. Estimation terminated at iteration number 7 because parameter estimates changed by less than ,001.

d. Estimation terminated at iteration number 8 because parameter estimates changed by less than ,001.

Figure D.2: Nagelkerke R^2 for multivariate logistic regression

Classification Table[a]

			Predicted		
			hasErrors		Percentage
	Observed		0	1	Correct
Step 1	hasErrors	0	1761	0	100,0
		1	213	0	,0
	Overall Percentage				89,2
Step 2	hasErrors	0	1736	25	98,6
		1	134	79	37,1
	Overall Percentage				91,9
Step 3	hasErrors	0	1720	41	97,7
		1	83	130	61,0
	Overall Percentage				93,7
Step 4	hasErrors	0	1719	42	97,6
		1	77	136	63,8
	Overall Percentage				94,0
Step 5	hasErrors	0	1719	42	97,6
		1	64	149	70,0
	Overall Percentage				94,6
Step 6	hasErrors	0	1721	40	97,7
		1	61	152	71,4
	Overall Percentage				94,9
Step 7	hasErrors	0	1722	39	97,8
		1	61	152	71,4
	Overall Percentage				94,9
Step 8	hasErrors	0	1723	38	97,8
		1	57	156	73,2
	Overall Percentage				95,2
Step 9	hasErrors	0	1724	37	97,9
		1	58	155	72,8
	Overall Percentage				95,2

a. The cut value is ,500

Figure D.3: Classification table for multivariate logistic regression

Variables in the Equation

		B	S.E.	Wald	df	Sig.	Exp(B)
Step 1[a]	Structuredness	-2,688	,093	843,418	1	,000	,068
Step 2[b]	N	,084	,005	247,455	1	,000	1,088
	Structuredness	-5,466	,237	530,121	1	,000	,004
Step 3[c]	N	,053	,006	73,718	1	,000	1,054
	Structuredness	-7,270	,387	353,553	1	,000	,001
	cHeterogeneity	4,419	,398	123,375	1	,000	83,029
Step 4[d]	N	,054	,006	73,082	1	,000	1,056
	CYC	4,392	,831	27,915	1	,000	80,835
	Structuredness	-7,495	,409	335,352	1	,000	,001
	cHeterogeneity	4,364	,411	112,589	1	,000	78,600
Step 5[e]	N	,043	,007	40,881	1	,000	1,044
	CNC	3,404	,712	22,878	1	,000	30,070
	CYC	3,995	,862	21,484	1	,000	54,342
	Structuredness	-10,333	,748	190,748	1	,000	,000
	cHeterogeneity	3,244	,457	50,273	1	,000	25,629
Step 6[f]	N	,039	,007	31,900	1	,000	1,040
	CNC	3,320	,708	22,013	1	,000	27,654
	MM	,067	,026	6,560	1	,010	1,069
	CYC	4,264	,873	23,857	1	,000	71,071
	Structuredness	-10,217	,744	188,622	1	,000	,000
	cHeterogeneity	2,778	,491	32,029	1	,000	16,084
Step 7[g]	N	,033	,007	21,363	1	,000	1,034
	CNC	3,898	,738	27,906	1	,000	49,285
	MM	,069	,025	7,407	1	,006	1,072
	CYC	3,825	,890	18,466	1	,000	45,852
	Separability	-1,648	,670	6,059	1	,014	,192
	Structuredness	-9,869	,757	169,882	1	,000	,000
	cHeterogeneity	2,723	,490	30,895	1	,000	15,222
Step 8[h]	N	,016	,011	1,946	1	,163	1,016
	CNC	3,805	,753	25,543	1	,000	44,919
	MM	,081	,026	9,670	1	,002	1,085
	CYC	3,601	,900	16,028	1	,000	36,642
	Separability	-1,980	,712	7,738	1	,005	,138
	Structuredness	-9,893	,760	169,376	1	,000	,000
	cHeterogeneity	2,882	,505	32,605	1	,000	17,849
	diameter	,041	,021	3,867	1	,049	1,042
Step 9[h]	CNC	4,008	,742	29,193	1	,000	55,033
	MM	,094	,025	14,572	1	,000	1,098
	CYC	3,409	,891	14,648	1	,000	30,248
	Separability	-2,338	,673	12,058	1	,001	,096
	Structuredness	-9,957	,760	171,551	1	,000	,000
	cHeterogeneity	3,003	,501	35,988	1	,000	20,139
	diameter	,064	,013	24,474	1	,000	1,066

a. Variable(s) entered on step 1: Structuredness.

b. Variable(s) entered on step 2: N.

c. Variable(s) entered on step 3: cHeterogeneity.

d. Variable(s) entered on step 4: CYC.

e. Variable(s) entered on step 5: CNC.

f. Variable(s) entered on step 6: MM.

g. Variable(s) entered on step 7: Separability.

h. Variable(s) entered on step 8: diameter.

Figure D.4: Equation of multivariate logistic regression models

D.4 Second Best Logistic Regression

After excluding the metrics of the regression model of Section D.3, i.e. without the coefficient of network connectivity CNC, connector mismatch MM, cyclicity CYC, separability Π, structuredness Φ, connector heterogeneity CH, and without the diameter $diam$, we calculated a second best multivariate logistic regression models. This model includes sequentiality Ξ, density Δ, and size S_N. Figure D.5 shows that the Hosmer & Lemeshow Test fails to indicate a good fit since the value is less than 5% after the second model. Figure D.6 summarizes the value of Nagelkerke's R^2 that indicates still a high fraction of explanation of the variability with a value of 0.824. Figure D.7 and D.8 give the classification tables and the equations of the models in the different steps.

Hosmer and Lemeshow Test

Step	Chi-square	df	Sig.
1	11,389	7	,123
2	92,939	8	,000
3	18,614	8	,017

Figure D.5: Hosmer and Lemeshow test for second best multivariate logistic regression

Model Summary

Step	-2 Log likelihood	Cox & Snell R Square	Nagelkerke R Square
1	1071,748[a]	,570	,760
2	945,296[a]	,596	,795
3	835,472[b]	,618	,824

a. Estimation terminated at iteration number 7 because parameter estimates changed by less than ,001.

b. Estimation terminated at iteration number 8 because parameter estimates changed by less than ,001.

Figure D.6: Nagelkerke R^2 for second best multivariate logistic regression

Classification Table[a]

			Predicted		
			hasErrors		Percentage
	Observed		0	1	Correct
Step 1	hasErrors	0	1703	58	96,7
		1	204	9	4,2
	Overall Percentage				86,7
Step 2	hasErrors	0	1761	0	100,0
		1	213	0	,0
	Overall Percentage				89,2
Step 3	hasErrors	0	1725	36	98,0
		1	134	79	37,1
	Overall Percentage				91,4

a. The cut value is ,500

Figure D.7: Classification table for second best multivariate logistic regression

Variables in the Equation

		B	S.E.	Wald	df	Sig.	Exp(B)
Step 1[a]	Sequentiality	-7,038	,315	498,244	1	,000	,001
Step 2[b]	Sequentiality	-3,596	,413	75,916	1	,000	,027
	Density	-20,822	2,327	80,046	1	,000	,000
Step 3[c]	Sequentiality	-6,540	,594	121,362	1	,000	,001
	Density	-23,873	2,590	84,992	1	,000	,000
	N	,034	,004	87,631	1	,000	1,034

a. Variable(s) entered on step 1: Sequentiality.

b. Variable(s) entered on step 2: Density.

c. Variable(s) entered on step 3: N.

Figure D.8: Equation of multivariate second best logistic regression models

D.5 Third Best Logistic Regression

After excluding the metrics of the regression model of Sections D.3 and D.4, i.e. only with token split TS, average and maximum connector degree $\overline{d_C}$ and $\widehat{d_C}$, and Depth Λ, we calculated a third best multivariate logistic regression models. Figure D.9 shows that the Hosmer & Lemeshow Test fails to indicate a good fit since the value is less than 5% after the second model. Figure D.10 summarizes the value of Nagelkerke's R^2 that indicates still a high fraction of explanation of the variability with a value of 0.627. Figure D.11 and D.12 give the classification tables and the equations of the models in the different steps.

Hosmer and Lemeshow Test

Step	Chi-square	df	Sig.
1	528,875	6	,000
2	389,011	7	,000
3	376,036	7	,000
4	363,645	7	,000

Figure D.9: Hosmer and Lemeshow test for third best multivariate logistic regression

Model Summary

Step	-2 Log likelihood	Cox & Snell R Square	Nagelkerke R Square
1	1793,593[a]	,380	,506
2	1529,029[b]	,458	,610
3	1496,768[b]	,466	,622
4	1481,988[b]	,470	,627

a. Estimation terminated at iteration number 4 because parameter estimates changed by less than ,001.

b. Estimation terminated at iteration number 5 because parameter estimates changed by less than ,001.

Figure D.10: Nagelkerke R^2 for third best multivariate logistic regression

Classification Table[a]

Observed			Predicted		
			hasErrors		Percentage Correct
			0	1	
Step 1	hasErrors	0	1414	347	80,3
		1	213	0	,0
	Overall Percentage				71,6
Step 2	hasErrors	0	1390	371	78,9
		1	164	49	23,0
	Overall Percentage				72,9
Step 3	hasErrors	0	1385	376	78,6
		1	164	49	23,0
	Overall Percentage				72,6
Step 4	hasErrors	0	1385	376	78,6
		1	159	54	25,4
	Overall Percentage				72,9

a. The cut value is ,500

Figure D.11: Classification table for third best multivariate logistic regression

Variables in the Equation

		B	S.E.	Wald	df	Sig.	Exp(B)
Step 1[a]	AvCDegree	-,532	,021	615,699	1	,000	,588
Step 2[b]	tokenSplit	,319	,024	183,587	1	,000	1,376
	AvCDegree	-,814	,033	602,859	1	,000	,443
Step 3[c]	tokenSplit	,222	,028	64,643	1	,000	1,248
	AvCDegree	-1,294	,093	195,393	1	,000	,274
	MaxCDegree	,425	,073	33,848	1	,000	1,530
Step 4[d]	tokenSplit	,194	,029	44,607	1	,000	1,214
	AvCDegree	-1,371	,097	200,313	1	,000	,254
	MaxCDegree	,405	,074	29,983	1	,000	1,500
	Depth	,440	,115	14,562	1	,000	1,553

a. Variable(s) entered on step 1: AvCDegree.

b. Variable(s) entered on step 2: tokenSplit.

c. Variable(s) entered on step 3: MaxCDegree.

d. Variable(s) entered on step 4: Depth.

Figure D.12: Equation of third best multivariate logistic regression models

Bibliography

[AAB+05] A. Arkin, S. Askary, B. Bloch, F. Curbera, Y. Goland, N. Kartha, C.K. Liu, S. Thatte, P. Yendluri, and A. Yiu. Web services business process execution language version 2.0. wsbpel-specification-draft-01, OASIS, September 2005.

[Aal97] W.M.P. van der Aalst. Verification of Workflow Nets. In Pierre Azéma and Gianfranco Balbo, editors, *Application and Theory of Petri Nets 1997*, volume 1248 of *Lecture Notes in Computer Science*, pages 407–426. Springer Verlag, 1997.

[Aal98] W.M.P. van der Aalst. The Application of Petri Nets to Workflow Management. *The Journal of Circuits, Systems and Computers*, 8(1):21–66, 1998.

[Aal99] W.M.P. van der Aalst. Formalization and Verification of Event-driven Process Chains. *Information and Software Technology*, 41(10):639–650, 1999.

[Aal03] W. M. P. van der Aalst. Patterns and XPDL: A Critical Evaluation of the XML Process Definition Language. QUT Technical report FIT-TR-2003-06, Queensland University of Technology, Brisbane, 2003.

[ABCC05] W.M.P. van der Aalst, B. Benatallah, F. Casati, and F. Curbera, editors. *Business Process Management, 3rd International Conference, BPM 2005, Nancy, France, September 5-8, 2005, Proceedings*, volume 3649, 2005.

[ACD+99] R. Anupindi, S. Chopra, S.D. Deshmukh, J.A. van Mieghem, and E. Zemel. *Managing Business Process Flows*. Prentice Hall, 1999.

[ACD+03] T. Andrews, F. Curbera, H. Dholakia, Y. Goland, J. Klein, F. Leymann, K. Liu, D. Roller, D. Smith, S. Thatte, I. Trickovic, and S. Weerawarana. Business Process Execution Language for Web Services, Version 1.1. Specification, BEA Systems, IBM Corp., Microsoft Corp., SAP AG, Siebel Systems, 2003.

[ADH+03] W.M.P. van der Aalst, B.F. van Dongen, J. Herbst, L. Maruster, G. Schimm, and A.J.M.M. Weijters. Workflow Mining: A Survey of Issues and Approaches. *Data & Knowledge Engineering*, 47(2):237–267, 2003.

[ADK02] W.M.P. van der Aalst, J. Desel, and E. Kindler. On the semantics of EPCs: A vicious circle. In M. Nüttgens and F.J. Rump, editor, *Proc. of the 1st GI-Workshop on Business Process Management with Event-Driven Process Chains (EPK 2002), Trier, Germany*, pages 71–79, 2002.

[ADO00] W.M.P. van der Aalst, J. Desel, and A. Oberweis, editors. *Business Process Management, Models, Techniques, and Empirical Studies*, volume 1806 of *Lecture Notes in Computer Science*. Springer, 2000.

[AGRP07] E. Rolón Aguilar, F. García, F. Ruiz, and M. Piattini. An exploratory experiment to validate measures for business process models. In *First International Conference on Research Challenges in Information Science (RCIS)*, 2007.

[AH02] W.M.P. van der Aalst and K. van Hee. *Workflow Management: Models, Methods, and Systems*. The MIT Press, 2002.

[AH05] W.M.P. van der Aalst and A.H.M. ter Hofstede. YAWL: Yet Another Workflow Language. *Information Systems*, 30(4):245–275, 2005.

[AHKB03] W.M.P. van der Aalst, A.H.M. ter Hofstede, B. Kiepuszewski, and A.P. Barros. Workflow Patterns. *Distributed and Parallel Databases*, 14(1):5–51, July 2003.

[AHV02] W.M.P. van der Aalst, A. Hirnschall, and H.M.W. Verbeek. An Alternative Way to Analyze Workflow Graphs. In A. Banks-Pidduck, J. Mylopoulos, C.C. Woo, and M.T. Ozsu, editors, *Proceedings of the 14th International Conference*

on Advanced Information Systems Engineering (CAiSE'02), volume 2348 of *LNCS*, pages 535–552. Springer-Verlag, Berlin, 2002.

[AHW03] W.M.P. van der Aalst, A.H.M. ter Hofstede, and M. Weske, editors. *Business Process Management, International Conference, BPM 2003, Eindhoven, The Netherlands, June 26-27, 2003, Proceedings*, volume 2678 of *Lecture Notes in Computer Science*. Springer, 2003.

[AJL05] W.M.P. van der Aalst, J.B. Jørgensen, and K.B. Lassen. Let's Go All the Way: From Requirements via Colored Workflow Nets to a BPEL Implementation of a New Bank System Paper. In R. Meersman and Z. Tari et al., editors, *On the Move to Meaningful Internet Systems 2005: CoopIS, DOA, and ODBASE: OTM Confederated International Conferences, CoopIS, DOA, and ODBASE 2005*, volume 3760 of *LNCS*, pages 22–39. Springer-Verlag, Berlin, 2005.

[AK01a] C. Atkinson and T. Kühne. The essence of multilevel metamodeling. In M. Gogolla and C. Kobryn, editors, *UML 2001 - The Unified Modeling Language, Modeling Languages, Concepts, and Tools, 4th International Conference, Toronto, Canada, October 1-5, 2001, Proceedings*, volume 2185 of *Lecture Notes in Computer Science*, pages 19–33. Springer, 2001.

[AK01b] C. Atkinson and T. Kühne. Processes and products in a multi-level metamodeling architecture. *International Journal of Software Engineering and Knowledge Engineering*, 11(6):761–783, 2001.

[AK03] C. Atkinson and T. Kühne. Model-driven development: A metamodeling foundation. *IEEE Software*, 20(5):36–41, 2003.

[AL05] W.M.P. van der Aalst and K.B. Lassen. Translating Workflow Nets to BPEL4WS. BETA Working Paper Series, WP 145, Eindhoven University of Technology, Eindhoven, 2005.

[Alb79] A.J. Albrecht. Measuring application development productivity. In *Proceeding IBM Applications Development Symposium, GUIDE Int and Share Inc., IBM Corp., Monterey, CA, Oct. 1417, 1979*, page 83, 1979.

[ARD+06] W.M.P. van der Aalst, V. Rubin, B.F. van Dongen, E. Kindler, and C.W. Günther. Process mining: A two-step approach using transition systems and regions. BPMCenter Report BPM-06-30, BPMcenter.org, 2006.

[ARGP06a] E. Rolón Aguilar, F. Ruiz, F. García, and M. Piattini. Applying software metrics to evaluate business process models. *CLEI Electron. J.*, 9, 2006.

[ARGP06b] E. Rolón Aguilar, F. Ruiz, F. García, and M. Piattini. Evaluation measures for business process models. In H. Haddad, editor, *Proceedings of the 2006 ACM Symposium on Applied Computing (SAC), Dijon, France, April 23-27, 2006*, pages 1567–1568. ACM, 2006.

[ARGP06c] E. Rolón Aguilar, F. Ruiz, F. García, and M. Piattini. Towards a Suite of Metrics for Business Process Models in BPMN. In Y. Manolopoulos, J. Filipe, P. Constantopoulos, and J. Cordeiro, editors, *ICEIS 2006 - Proceedings of the Eighth International Conference on Enterprise Information Systems: Databases and Information Systems Integration (III), Paphos, Cyprus, May 23-27, 2006*, pages 440–443, 2006.

[AS02] A.D. Aczel and J. Sounderpandian. *Complete Business Statistics*. McGraw-Hill, 5th edition, 2002.

[AS07] T. Ami and R. Sommer. Comparison and evaluation of business process modelling and management tools. *International Journal of Services and Standards,*, 3(2):249261, 2007.

[ATM03] O. Adam, O. Thomas, and G. Martin. Fuzzy enhanced process management for the industrial order handling. In A.-W. Scheer, editor, *Proceedings of the 5th International Conference ; The Modern Information Technology in the Innovation Processes of the Industrial Enterprises, MITIP 2003*, number 176 in Veröffentlichungen des Instituts für Wirtschaftsinformatik, pages 15–20. German Research Center for Artificial Intelligence, September 2003.

[Aus62] J. L. Austin. *How to Do Things with Words*. Harvard University Press, Cambridge, Mass., 1962.

[AWM04] W.M.P. van der Aalst, A.J.M.M. Weijters, and L. Maruster. Workflow Mining: Discovering Process Models from Event Logs. *IEEE Transactions on Knowledge and Data Engineering*, 16(9):1128–1142, 2004.

[BA99] A.-L. Barabási and R. Albert. Emergence of scaling in random networks. *Science*, 286:509–512, 1999.

[Bal98] H. Balzert. *Lehrbuch der Software-Technik: Software-Management, Software-Qualitätssicherung, Unternehmensmodellierung*. Spektrum Akademischer Verlag, 1998.

[BAN03] J. Becker, L. Algermissen, and B. Niehaves. Prozessmodellierung in eGovernment-Projekten mit der eEPK. In M. Nüttgens, F.J. Rump, editor, *Proc. of the 2nd GI-Workshop on Business Process Management with Event-Driven Process Chains (EPK 2003), Bamberg, Germany*, pages 31–44, 2003.

[BBC+04] P. Bernstein, A. Borgida, M. Carey, S. Chaudhuri, P. Chrysanthis, C. Cliftonand U. Dayal, D. Florescu, M. Franklin, L. Gasser, J. Gehrke, S. Ghandeharizadeh, A. Halevy, H.V. Jagadish, H. Korth, R. Ramakrishnan, A. Rosenthal, S. Sripada, J. Srivastava, J. Su, D. Suciu, S. Thatte, B. Thuraisingham, V. Tsotras, J. Widom, M. Winslett, O. Wolfson, J. Yuan, and M. Zemankova. Science of design for information systems: report of the NSF workshop, Seattle, 2003. *SIGMOD Record*, 33(1):133–137, 2004.

[BD98] E. Badouel and P. Darondeau. Theory of regions. In W. Reisig and G. Rozenberg, editors, *Lectures on Petri Nets I: Basic Models, Advances in Petri Nets, the volumes are based on the Advanced Course on Petri Nets, held in Dagstuhl, September 1996*, volume 1491 of *Lecture Notes in Computer Science*, pages 529–586. Springer, 1998.

[BE05a] U. Brandes and T. Erlebach. Fundamentals. In U. Brandes and T. Erlebach, editors, *Network Analysis: Methodological Foundations [outcome of a Dagstuhl seminar, 13-16 April 2004]*, volume 3418 of *Lecture Notes in Computer Science*, pages 7–15. Springer, 2005.

[BE05b] U. Brandes and T. Erlebach. Introduction. In U. Brandes and T. Erlebach, editors, *Network Analysis: Methodological Foundations [outcome of a Dagstuhl seminar, 13-16 April 2004]*, volume 3418 of *Lecture Notes in Computer Science*, pages 1–6. Springer, 2005.

[BE05c] U. Brandes and T. Erlebach, editors. *Network Analysis: Methodological Foundations [outcome of a Dagstuhl seminar, 13-16 April 2004]*, volume 3418 of *Lecture Notes in Computer Science*. Springer, 2005.

[BEPW03] K. Backhaus, B. Erichson, W. Plinke, and R. Weiber. *Multivariate Analysemethoden. Eine anwendungsorientierte Einführung*. Springer-Verlag, 10th edition, 2003.

[Ber86] G. Berthelot. Checking Properties of Nets Using Transformations. In G. Rozenberg, editor, *Advances in Petri Nets 1985*, volume 222 of *LNCS*, pages 19–40. Springer-Verlag, Berlin, 1986.

[Ber87] G. Berthelot. Transformations and Decompositions of Nets. In W. Brauer, W. Reisig, and G. Rozenberg, editors, *Advances in Petri Nets 1986 Part I: Petri Nets, central models and their properties*, volume 254 of *LNCS*, pages 360–376. Springer-Verlag, Berlin, 1987.

[BG05] S. Balasubramanian and M. Gupta. Structural metrics for goal based business process design and evaluation. *Business Process Management Journal*, 11(6):680–694, 2005.

[BH05] H.U. Buhl and B. Heinrich. Meinung/Dialog: Empirical Research Strategies in Conceptual Modeling - Silver Bullet or Academic Toys? *Wirtschaftsinformatik*, 47(2):152–162, 2005.

[BHK+06] P. Barborka, L. Helm, G. Köldorfer, J. Mendling, G. Neumann, B.F. van Dongen, H.M.W. Verbeek, and W.M.P. van der Aalst. Integration of EPC-related Tools with ProM. In M. Nüttgens and F.J. Rump and J. Mendling, editor, *Proceedings of the 5th GI Workshop on Business Process Management*

with Event-Driven Process Chains (EPK 2006), pages 105–120, Vienna, Austria, December 2006. German Informatics Society.

[BK03] J. Becker and M. Kugeler. *Process Management: A Guide for the Design of Business Processes*, chapter The Process in Focus, pages 1–12. Springer-Verlag, 2003.

[BKKR03] M. Bernauer, G. Kappel, G. Kramler, and W. Retschitzegger. Specification of Interorganizational Workflows - A Comparison of Approaches. In *Proceedings of the 7th World Multiconference on Systemics, Cybernetics and Informatics*, pages 30–36, 2003.

[BKR03] J. Becker, M. Kugeler, and M. Rosemann. *Process Management: A Guide for the Design of Business Processes*. Springer-Verlag, 2003.

[BLHL01] Tim Berners-Lee, James Hendler, and Ora Lassila. The semantic web. *Scientific American*, May 2001.

[BLN86] C. Batini, M. Lenzerini, and S. B. Navathe. A Comparative Analysis of Methodologies for Database Schema Integration. *ACM Computing Surveys*, 18(4):323–364, December 1986.

[BN07] J. Becker and B. Niehaves. Epistemological perspectives on is research - a framework for analyzing and systematizing epistemological assumptions. *Information Systems Journal*, 2007. to appear.

[BO02] E. Brabänder and J. Ochs. Analyse und Gestaltung prozessorientierter Risikomanagementsysteme mit Ereignisgesteuerten Prozessketten. In M. Nüttgens and F.J. Rump, editor, *Proc. of the 1st GI-Workshop on Business Process Management with Event-Driven Process Chains (EPK 2002), Trier, Germany*, pages 17–35, 2002.

[Boe79] B. W. Boehm. *Research Directions in Software Technology*, chapter Software engineering; R & D trends and defense needs. MIT Press, 1979.

[Boe81] B.W. Boehm. *Software Engineering Economics*. Prentice-Hall, Englewood Cliffs, 1981.

[BP84] V.R. Basili and B.T. Perricone. Software errors and complexity: An empirical investigation. *Communications of the ACM*, 27(1):42–52, 1984.

[BP98] K. Barkaoui and L. Petrucci. Structural Analysis of Workflow Nets with Shared Resources. In W.M.P. van der Aalst, G. De Michelis, and C.A. Ellis, editors, *Proceedings of Workflow Management: Net-based Concepts, Models, Techniques and Tools (WFM'98)*, volume 98/7 of *Computing Science Reports*, pages 82–95, Lisbon, Portugal, 1998. Eindhoven University of Technology, Eindhoven.

[BR88] V.R. Basili and H. Dieter Rombach. The TAME project: Towards improvement-oriented software environments. *IEEE Transactions on Software Engineering*, 14(6):758–773, 1988.

[Bro03] J. vom Brocke. *Referenzmodellierung - Gestaltung und Verteilung von Konstruktionsprozessen*. Number 4 in Advances in Information Systems and Management Science. Logos Verlag Berlin, 2003.

[BRS95] J. Becker, M. Rosemann, and R. Schütte. Grundsätze ordnungsmässiger Modellierung. *Wirtschaftsinformatik*, 37(5):435–445, 1995.

[BRU00] J. Becker, M. Rosemann, and C. von Uthmann. Guidelines of Business Process Modeling. In W.M.P. van der Aalst, J. Desel, and A. Oberweis, editors, *Business Process Management. Models, Techniques, and Empirical Studies*, pages 30–49. Springer, Berlin et al., 2000.

[BS04] J. Becker and R. Schütte. *Handelsinformationssysteme*. Moderne Industrie, Landsberg/Lech, 2nd edition, 2004.

[BS05] M. Brinkmeier and T. Schank. Network statistics. In U. Brandes and T. Erlebach, editors, *Network Analysis: Methodological Foundations [outcome of a Dagstuhl seminar, 13-16 April 2004]*, volume 3418 of *Lecture Notes in Computer Science*, pages 293–317. Springer, 2005.

[Bun77] M. Bunge. *Treatise on Basic Philosophy. Vol.3. Ontology I. The Furniture of the World*. D. Reidel Publishing, New York, 1977.

[BW84] V.R. Basili and D.M. Weiss. A methodology for collecting valid software engineering data. *IEEE Transactions on Software Engineering*, 10(6):728–738, 1984.

[Car05a] J. Cardoso. About the complexity of teamwork and collaboration processes. In *2005 IEEE/IPSJ International Symposium on Applications and the Internet Workshops (SAINT 2005 Workshops), 31 January - 4 February 2005, Trento, Italy*, pages 218–221. IEEE Computer Society, 2005.

[Car05b] J. Cardoso. Control-flow Complexity Measurement of Processes and Weyuker's Properties. In *6th International Enformatika Conference*, Transactions on Enformatika, Systems Sciences and Engineering, Vol. 8, pages 213–218, 2005.

[Car05c] J. Cardoso. Evaluating the process control-flow complexity measure. In *2005 IEEE International Conference on Web Services (ICWS 2005), 11-15 July 2005, Orlando, FL, USA*, pages 803–804. IEEE Computer Society, 2005.

[Car05d] J. Cardoso. *Workflow Handbook 2005*, chapter Evaluating Workflows and Web Process Complexity, pages 284–290. Future Strategies, Inc., Lighthouse Point, FL, USA, 2005.

[Car06] J. Cardoso. Process control-flow complexity metric: An empirical validation. In *Proceedings of IEEE International Conference on Services Computing (IEEE SCC 06), Chicago, USA, September 18-22*, pages 167–173. IEEE Computer Society, 2006.

[CCPP95] F. Casati, S. Ceri, B. Pernici, and G. Pozzi. Conceptual modeling of workflows. In *Proceedings of the OOER International Conference*, Gold Cost, Australia, 1995.

[CFK05] N. Cuntz, J. Freiheit, and E. Kindler. On the semantics of EPCs: Faster calculation for EPCs with small state spaces. In M. Nüttgens and F.J. Rump, editor,

Proceedings of the 4th GI Workshop on Business Process Management with Event-Driven Process Chains (EPK 2005), pages 7–23, Hamburg, Germany, December 2005. German Informatics Society.

[CGK+02] F. Curbera, Y. Goland, J. Klein, F. Leymann, D. Roller, S. Thatte, and S. Weerawarana. Business Process Execution Language for Web Services, Version 1.0. Specification, BEA Systems, IBM Corp., Microsoft Corp., 2002.

[CGP+05] G. Canfora, F. García, M. Piattini, F. Ruiz, and C.A. Visaggio. A family of experiments to validate metrics for software process models. *Journal of Systems and Software*, 77(2):113–129, 2005.

[Che76] P. Chen. The Entity-Relationship Model - Towards a Unified View of Data. *ACM Transactions on Database Systems (TODS)*, (1):9–36, 1976.

[CK94] S.R. Chidamber and C.F. Kemerer. A metrics suite for object oriented design. *IEEE Transaction on Software Engineering*, 20(6):476–493, 1994.

[CK04] N. Cuntz and E. Kindler. On the semantics of EPCs: Efficient calculation and simulation. In *Proceedings of the 3rd GI Workshop on Business Process Management with Event-Driven Process Chains (EPK 2004)*, pages 7–26, 2004.

[CK05] N. Cuntz and E. Kindler. On the semantics of epcs: Efficient calculation and simulation. In W.M.P. van der Aalst, B. Benatallah, F. Casati, and F. Curbera, editors, *Business Process Management, 3rd International Conference, BPM 2005, Nancy, France, September 5-8, 2005, Proceedings*, volume 3649 of *Lecture Notes in Computer Science*, pages 398–403, 2005.

[CKL97] T. Curran, G. Keller, and A. Ladd. *SAP R/3 Business Blueprint: Understanding the Business Process Reference Model*. Enterprise Resource Planning Series. Prentice Hall PTR, Upper Saddle River, 1997.

[CKLY98] J. Cortadella, M. Kishinevsky, L. Lavagno, and A. Yakovlev. Deriving petri nets from finite transition systems. *IEEE Transactions on Computers*, 47(8):859–882, August 1998.

[CLRS01] T.H. Cormen, C.E. Leiserson, R.L. Rivest, and C. Stein. *Introduction to Algorithms*. The MIT Press, 2nd edition, 2001.

[CMNR06] J. Cardoso, J. Mendling, G. Neumann, and H. Reijers. A Discourse on Complexity of Process Models. In Johann Eder and Schahram Dustdar, editors, *Proceedings of BPM Workshops 2006*, volume 4103 of *Lecture Notes in Computer Science*, pages 115–126, Vienna, Austria, 2006. Springer-Verlag.

[Cor98] J. Cortadella. *Petrify: a tutorial for the designer of asychronous circuits*. Universitat Politécnica de Catalunya, http://www.lsi.upc.es/petrify, December 1998.

[CS91] J.C. Cherniavsky and C.H. Smith. On weyuker's axioms for software complexity measures. *IEEE Transactions on Software Engineering*, 17(6):636–638, 1991.

[CS94] R. Chen and A. W. Scheer. Modellierung von Prozessketten mittels Petri-Netz-Theorie. Heft 107, Institut für Wirtschaftsinformatik, Saarbrücken, Germany, 1994.

[Cun04] N. Cuntz. Über die effiziente Simulation von Ereignisgesteuerten Prozessketten. Master's thesis, University of Paderborn, June 2004. (in German).

[CW98] J.E. Cook and A.L. Wolf. Discovering Models of Software Processes from Event-Based Data. *ACM Transactions on Software Engineering and Methodology*, 7(3):215–249, 1998.

[DA04] J. Dehnert and W.M.P. van der Aalst. Bridging The Gap Between Business Models And Workflow Specifications. *International J. Cooperative Inf. Syst.*, 13(3):289–332, 2004.

[Dav89] F.D. Davis. Perceived usefulness, perceived ease of use, and user acceptance of information technology. *MIS Quarterly*, 13(3):319–340, 1989.

[Dav93] T. H. Davenport. *Process Innovation: Reengineering Work Through Information Technology*. Harvard Business School Press, Boston, 1993.

[DAV05] B.F. van Dongen, W.M.P. van der Aalst, and H.M.W. Verbeek. Verification of EPCs: Using reduction rules and Petri nets. In O. Pastor and J. Falcão e Cunha, editors, *Advanced Information Systems Engineering, 17th International Conference, CAiSE 2005, Porto, Portugal, June 13-17, 2005, Proceedings*, volume 3520 of *Lecture Notes in Computer Science*, pages 372–386. Springer, 2005.

[DE95] J. Desel and J. Esparza. *Free Choice Petri Nets*, volume 40 of *Cambridge Tracts in Theoretical Computer Science*. Cambridge Univ. Press, Cambridge, UK, 1995.

[Deh02] J. Dehnert. Making EPCs fit for Workflow Management. In M. Nüttgens and F.J. Rump, editor, *Proc. of the 1st GI-Workshop on Business Process Management with Event-Driven Process Chains (EPK 2002), Trier, Germany*, pages 51–69, 2002.

[DeM82] T. DeMarco. *Controlling Software Projects*. Yourdon Press, New York, 1982.

[Des05] J. Desel. *Process Aware Information Systems: Bridging People and Software Through Process Technology*, chapter Process Modeling Using Petri Nets, pages 147–178. Wiley Publishing, 2005.

[DFS06] S. Dustdar, J.L. Fiadeiro, and A.P. Sheth, editors. *Business Process Management, 4th International Conference, BPM 2006, Vienna, Austria, September 5-7, 2006, Proceedings*, volume 4102 of *Lecture Notes in Computer Science*. Springer, 2006.

[DGR+06] I. Davies, P. Green, M. Rosemann, M. Indulska, and S. Gallo. How do practitioners use conceptual modeling in practice? *Data & Knowledge Engineering*, 58(3):358–380, 2006.

[DHA05] M. Dumas, A. ter Hofstede, and W.M.P. van der Aalst. *Process Aware Information Systems: Bridging People and Software Through Process Technology*, chapter Introduction, pages 3–20. Wiley Publishing, 2005.

[DHL01] U. Dayal, M. Hsu, and R. Ladin. Business Process Coordination: State of the Art, Trends, and Open Issues. In P.M.G. Apers and P. Atzeni and S. Ceri and S.

Paraboschi and K. Ramamohanarao and R.T. Snodgrass, editor, *Proc. of 27th International Conference on Very Large Data Bases (VLDB), Roma, Italy, Sept. 2001*, 2001.

[DJV05] Boudewijn F. van Dongen and M. H. Jansen-Vullers. Verification of SAP reference models. In W.M.P. van der Aalst, B. Benatallah, F. Casati, and F. Curbera, editors, *Business Process Management, 3rd International Conference, BPM 2005, Nancy, France, September 5-8, 2005, Proceedings*, volume 3649 of *Lecture Notes in Computer Science*, pages 464–469, 2005.

[DMA06] B.F. van Dongen, J. Mendling, and W.M.P. van der Aalst. Structural Patterns for Soundness of Business Process Models. In *Proceedings of EDOC 2006*, Hong Kong, China, 2006. IEEE.

[DMV+05] B.F. van Dongen, A.K. Alves de Medeiros, H.M.W. Verbeek, A.J.M.M. Weijters, and W.M.P. van der Aalst. The ProM framework: A New Era in Process Mining Tool Support. In G. Ciardo and P. Darondeau, editors, *Application and Theory of Petri Nets 2005*, volume 3536 of *LNCS*, pages 444–454. Springer-Verlag, Berlin, 2005.

[Don07] B.F. van Dongen. *Process Mining and Verification*. PhD thesis, Eindhoven University of Technology, Eindhoven, The Netherlands, 2007.

[DPW04] J. Desel, B. Pernici, and M. Weske, editors. *Business Process Management: Second International Conference, BPM 2004, Potsdam, Germany, June 17-18, 2004. Proceedings*, volume 3080 of *Lecture Notes in Computer Science*. Springer, 2004.

[DR01] J. Dehnert and P. Rittgen. Relaxed Soundness of Business Processes. In K.R. Dittrick, A. Geppert, and M.C. Norrie, editors, *Proceedings of the 13th International Conference on Advanced Information Systems Engineering*, volume 2068 of *Lecture Notes in Computer Science*, pages 151–170, Interlaken, 2001. Springer.

[DZ05] J. Dehnert and A. Zimmermann. On the suitability of correctness criteria for business process models. In W.M.P. van der Aalst, B. Benatallah, F. Casati, and F. Curbera, editors, *Business Process Management, 3rd International Conference, BPM 2005, Nancy, France, September 5-8, 2005, Proceedings*, volume 3649 of *Lecture Notes in Computer Science*, pages 386–391, 2005.

[Ell79] C.A. Ellis. Information Control Nets: A Mathematical Model of Office Information Flow. In *Proceedings of the Conference on Simulation, Measurement and Modeling of Computer Systems*, pages 225–240, Boulder, Colorado, 1979. ACM Press.

[EN80] C.A. Ellis and G.J. Nutt. Office information systems and computer science. *ACM Computing Surveys*, 12(1):27–60, 1980.

[EN93] C.A. Ellis and G.J. Nutt. Modelling and Enactment of Workflow Systems. In M. Ajmone Marsan, editor, *Application and Theory of Petri Nets 1993*, volume 691 of *LNCS*, pages 1–16. Springer-Verlag, Berlin, 1993.

[ER89] A. Ehrenfeucht and G. Rozenberg. Partial (Set) 2-Structures - Part 1 and Part 2. *Acta Informatica*, 27(4):315–368, 1989.

[ER05] A. Etien and C. Rolland. Measuring the fitness relationship. *Requir. Eng.*, 10(3):184–197, 2005.

[Esp94] J. Esparza. Reduction and synthesis of live and bounded free choice petri nets. *Information and Computation*, 114(1):50–87, 1994.

[Fay66] H. Fayol. *Administration industrielle et générale. Prévoyance, Organisation, Commandement, Coordination, Control*. Dunod, 1966.

[Fet06] P. Fettke. *Referenzmodellevaluation - Konzeption der strukturalistischen Referenzmodellierung und Entfaltung ontologischer Gütekriterien*. PhD thesis, Universität des Saarlandes, 2006.

[FJJ03] J. Ferraiolo, F. Jun, and D. Jackson. Scalable Vector Graphics (SVG) 1.1. W3C Recommendation 14 January 2003, World Wide Web Consortium, 2003.

[FL03] P. Fettke and P. Loos. Classification of reference models - a methodology and its application. *Information Systems and e-Business Management*, 1(1):35–53, 2003.

[Fla98] R.G. Flatscher. *Meta-Modellierung in EIA/CDIF*. ADV-Verlag, Wien, 1998.

[FO00] N. E. Fenton and N. Ohlsson. Quantitative analysis of faults and failures in a complex software system. *IEEE Transactions on Software Engineering*, 26(8):797–814, August 2000.

[For26] H. Ford. *Today and Tomorrow*. Doubleday, Page and Company, 1926.

[FP97] N.E. Fenton and S.L. Pfleeger. *Software Metrics. A Rigorous and Practical Approach*. PWS, Boston, 1997.

[Fra99] U. Frank. Conceptual Modelling as the Core of Information Systems Discipline - Perspectives and Epistemological Challenges. In *Proceedings of the America Conference on Information Systems - AMCIS '99*, pages 695–698, Milwaukee, 1999.

[Fre79] L.C. Freeman. Centrality in social networks. conceptual clarification. *Social Networks*, 1(3):215–239, 1979.

[FS98] O.K. Ferstl and E.J. Sinz. *Grundlagen der Wirtschaftsinformatik Band 1*. R. Oldenbourg Verlag, 3rd edition, 1998.

[FS01] A. Finkel and Ph. Schnoebelen. Well-structured Transition Systems everywhere! *Theoretical Computer Science*, 256(1–2):63–92, April 2001.

[FW06] P.J.M. Frederiks and T.P. van der Weide. Information modeling: The process and the required competencies of its participants. *Data & Knowledge Engineering*, 58(1):4–20, 2006.

[GCC+04] D. Grigori, F. Casati, M. Castellanos, U. Dayal, M. Sayal, and M.C. Shan. Business Process Intelligence. *Computers in Industry*, 53(3):321–343, 2004.

[GD05a] A. Selçuk Güceglioglu and O. Demirörs. A process based model for measuring process quality attributes. In Ita Richardson, Pekka Abrahamsson, and Richard Messnarz, editors, *Software Process Improvement, 12th European Conference, EuroSPI 2005, Budapest, Hungary, November 9-11, 2005, Proceedings*, volume 3792 of *Lecture Notes in Computer Science*, pages 118–129. Springer, 2005.

[GD05b] A. Selçuk Güceglioglu and O. Demirörs. Using software quality characteristics to measure business process quality. In W.M.P. van der Aalst, B. Benatallah, F. Casati, and F. Curbera, editors, *Business Process Management, 3rd International Conference, BPM 2005, Nancy, France, September 5-8, 2005, Proceedings*, volume 3649 of *Lecture Notes in Computer Science (LNCS)*, pages 374–379. Springer Verlag, 2005.

[GH03] J.F. Groote and F. van Ham. Large state space visualization. In John Hatcliff Hubert Garavel, editor, *Tools and Algorithms for the Construction and Analysis of Systems, 9th International Conference, TACAS 2003, Held as Part of the Joint European Conferences on Theory and Practice of Software, ETAPS 2003*, volume 2619 of *Lecture Notes in Computer Science*, pages 585–590. Springer-Verlag, 2003.

[GH06] J.F. Groote and F. van Ham. Interactive visualization of large state spaces. *International Journal on Software Tools for Technology Transfer*, 8(1):77–91, 2006.

[GHS95] D. Georgakopoulos, M. Hornick, and A. Sheth. An Overview of Workflow Management: From Process Modeling to Workflow Automation Infrastructure. *Distributed and Parallel Databases*, 3:119–153, 1995.

[GHW02] G. Guizzardi, H. Herre, and G. Wagner. On the general ontological foundations of conceptual modeling. In S. Spaccapietra, S.T. March, and Y. Kambayashi, editors, *Conceptual Modeling - ER 2002, 21st International Conference on Conceptual Modeling, Tampere, Finland, October 7-11, 2002, Pro-*

ceedings, volume 2503 of *Lecture Notes in Computer Science*, pages 65–78. Springer, 2002.

[Gil88] T. Gilb. *Principles of Software Engineering Management*. Addison-Wesley, Reading, MA, 1988.

[GJM03] C. Ghezzi, M. Jazayeri, and D. Mandrioli. *Fundamentals of Software Engineering*. Prentice Hall, Pearson Education, Upper Saddle River, 2nd edition, 2003.

[GK91] G.K. Gill and C.F. Kemerer. Cyclomatic complexity density and software maintenance productivity. *IEEE Transaction on Software Engineering*, 17(9):1284–1288, 1991.

[GL07] Volker Gruhn and Ralf Laue. What business process modelers can learn from programmers. *Science of Computer Programming*, 65(1):4–13, 2007.

[GLM06] V. Gruhn, R. Laue, and F. Meyer. Berechnung von Komplexitätsmetriken für ereignisgesteuerte Prozessketten. In M. Nüttgens and F.J. Rump and J. Mendling, editor, *Proceedings of the 5th GI Workshop on Business Process Management with Event-Driven Process Chains (EPK 2006)*, pages 189–202, Vienna, Austria, December 2006. German Informatics Society.

[Gon86] G. Gong. Cross-validation, the jackknife, and the bootstrap: Excess error estimation in forward logistic regression. *Journal of the American Statistical Association*, 81(393):108–113, March 1986.

[Gro66] E. Grochla. *Automation und Organisation*. Betriebs-wirtschaftlicher Verlag Dr. Th. Gabler, 1966.

[Gro68] E. Grochla. Die Integration der Datenverarbeitung. Durchführung anhand eines integrierten Unternehmensmodells. *Bürotechnik und Automation*, 9:108–120, 1968.

[Gro74] E. Grochla. *Integrierte Gesamtmodelle der Datenverarbeitung. Entwicklung und Anwendung des Kölner Integrationsmodells (KIM)*. Hanser, München - Wien, 1974.

[Gro75] E. Grochla. *Betriebliche Planung und Informationssysteme*. Rowohlt, 1975.

[GRSS05] G. Grossmann, Y. Ren, M. Schrefl, and M. Stumptner. Behavior based integration of composite business processes. In W.M.P. van der Aalst, B. Benatallah, F. Casati, and F. Curbera, editors, *Business Process Management, 3rd International Conference, BPM 2005, Nancy, France, September 5-8, 2005, Proceedings*, volume 3649 of *Lecture Notes in Computer Science*, pages 186–204. Springer, 2005.

[GS75] E. Grochla and N. Szyperski. *Information Systems and Organizational Structure*. Walter de Gruyter, 1975.

[GS95] J. Galler and A.-W. Scheer. Workflow-Projekte: Vom Geschäftsprozeßmodell zur unternehmensspezifischen Workflow-Anwendung. *Information Management*, 10(1):20–27, 1995.

[Gut77] L. Guttman. What is not what in statistics. *The statistician*, 26:81–107, 1977.

[GW03] A. Gemino and Y. Wand. Evaluating modeling techniques based on models of learning. *Commun. ACM*, 46(10):79–84, 2003.

[GW04] A. Gemino and Y. Wand. A framework for empirical evaluation of conceptual modeling techniques. *Requir. Eng.*, 9(4):248–260, 2004.

[GW05] A. Gemino and Y. Wand. Complexity and clarity in conceptual modeling: Comparison of mandatory and optional properties. *Data & Knowledge Engineering*, 55(3):301–326, 2005.

[Hal77] M. H. Halstead. *Elements of Software Science.*, volume 7 of *Operating, and Programming Systems Series*. Elsevier, Amsterdam, 1977.

[Han70] H.R. Hansen. *Bestimmungsfaktoren fü den Einsatz elektronischer Datenverarbeitungsanlagen in Unternehmungen*. Betriebspolitische Schriften - Beiträge zur Unternehmenspolitik. Duncker & Humblot, Berlin, 1970.

[Har94] A. Hars. *Referenzdatenmodelle. Grundlagen effizienter Datenmodellierung*. Gabler Verlag, 1994.

[HATB98] J. F. Hair, jr., R. E. Anderson, R. L. Tatham, and W. C. Black. *Multivariate Data Analysis*. Prentice-Hall International, Inc., 5th edition edition, 1998.

[Hav05] Mike Havey. *Essential Business Process Modeling*. O'Reilly, August 2005.

[HC93] M. Hammer and J. Champy. *Reengineering the Corporation: A Manifesto for Business Revolution*. Harpercollins, New York, 1993.

[Hei96] H. Heilmann. *Information Engineering*, chapter Die Integration der Aufbauorganisation in Workflow-Management-Systemen, pages 147–165. 1996.

[Her00] J. Herbst. A Machine Learning Approach to Workflow Management. In *Proceedings 11th European Conference on Machine Learning*, volume 1810 of *LNCS*, pages 183–194. Springer-Verlag, Berlin, 2000.

[HFKV06] R. Hauser, M. Fries, J.M. Küster, and J. Vanhatalo. Combining analysis of unstructured workflows with transformation to structured workflows. In *Proceedings of EDOC 2006*. IEEE Publishing, October 2006.

[HHK06] B. Hofreiter, C. Huemer, and J.-H. Kim. Choreography of ebxml business collaborations. *Information Systems and E-Business Management*, 2006.

[HHR07] L.J. Heinrich, A. Heinzl, and F. Roithmayr. *Wirtschaftsinformatik. Einfürung und Grundlegung*. R. Oldenbourg Verlag, 3rd edition, 2007.

[HK81] S. Henry and D. Kafura. Software structure metrics based on information-flow. *IEEE Transactions On Software Engineering*, 7(5):510–518, 1981.

[HK89] Rudy Hirschheim and Heinz K. Klein. Four paradigms of information systems development. *Commun. ACM*, 32(10):1199–1216, 1989.

[HK96] M. Hsu and C. Kleissner. Objectflow: Towards a process management infrastructure. *Distributed and Parallel Databases*, 4(2):169–194, 1996.

[HL00] D.W. Hosmer and S. Lemeshow. *Applied Logistic Regression*. John Wiley & Sons, 2nd edition, 2000.

[HMPR04] A.R. Hevner, S.T. March, J. Park, and S. Ram. Design science in information systems research. *MIS Quarterly*, 28(1):75–105, March 2004.

[HN05] H.R. Hansen and G. Neumann. *Wirtschaftsinformatik 1: Grundlagen und Anwendungen*. Lucius & Lucius, 9th edition, 2005.

[Hol94] D. Hollingsworth. The Workflow Reference Model. TC00-1003 Issue 1.1, Workflow Management Coalition, 24 November 1994.

[Hol04] D. Hollingsworth. *The Workflow Handbook 2004*, chapter The Workflow Reference Model: 10 Years On, pages 295–312. Workflow Management Coalition, 2004.

[HOS05] K. van Hee, O. Oanea, and N. Sidorova. Colored Petri Nets to Verify Extended Event-Driven Process Chains. In R. Meersman and Z. Tari, editors, *Proceedings of CoopIS/DOA/ODBASE 2005*, volume 3760 of *Lecture Notes in Computer Science*, pages 183–201. Springer-Verlag, 2005.

[HOS+06] K. van Hee, O. Oanea, A. Serebrenik, N. Sidorova, and M. Voorhoeve. Workflow model compositions perserving relaxed soundness. In S. Dustdar, J.L. Fiadeiro, and A. Sheth, editors, *Business Process Management, 4th International Conference, BPM 2006*, volume 4102 of *Lecture Notes in Computer Science*, pages 225–240. Springer-Verlag, September 2006.

[HOSV06] K. van Hee, O. Oanea, N. Sidorova, and M. Voorhoeve. Verifying generalized soundness for workflow nets. In I. Virbitskaite and A. Voronkov, editors, *Proc. of the 6th International Conference on Perspectives of System Informatics, PSI'2006*, volume 4378 of *Lecture Notes in Computer Science*, pages 231–244, Novosibirsk, June 2006. Springer-Verlag.

[HPW05] S. Hoppenbrouwers, H.A. Proper, and T.P. van der Weide. A fundamental view on the process of conceptual modeling. In L.M.L. Delcambre, C. Kop, H.C. Mayr, J. Mylopoulos, and O. Pastor, editors, *Conceptual Modeling - ER 2005, 24th International Conference on Conceptual Modeling, Klagenfurt, Austria,*

October 24-28, 2005, Proceedings, volume 3716 of *Lecture Notes in Computer Science*, pages 128–143. Springer, 2005.

[HR07] M. Hepp and D. Roman. An ontology framework for semantic business process management. In A. Oberweis, C. Weinhardt, H. Gimpel, A. Koschmider, V. Pankratius, and B. Schnizler, editors, *eOrganisation: Service-, Prozess-, Market Engineering*, pages 423–440. Universitätsverlag Karlsruhe, 2007. Tagungsband der 8. Internationalen Tagung Wirtschaftsinformatik. Band 1.

[HSSV06] K. van Hee, N. Sidorova, L. Somers, and M. Voorhoeve. Consistency in model integration. *Data & Knowledge Engineering*, 56, 2006.

[HSV04] K. van Hee, N. Sidorova, and M. Voorhoeve. Generalised Soundness of Workflow Nets Is Decidable. In J. Cortadella and W. Reisig, editors, *Application and Theory of Petri Nets 2004*, volume 3099 of *LNCS*, pages 197–215. Springer-Verlag, Berlin, 2004.

[HT74] J.E. Hopcroft and R.E. Tarjan. Efficient planarity testing. *Journal of the ACM*, 21(4):549–568, 1974.

[HWS93] W. Hoffmann, R. Wein, and A.-W. Scheer. Konzeption eines Steuerungsmodelles für Informationssysteme - Basis für Real-Time-Erweiterung der EPK (rEPK). Heft 106, Institut für Wirtschaftsinformatik, Saarbrücken, Germany, 1993.

[HWW02] F. van Ham, H. van de Wetering, and J.J. van Wijk. Interactive visualization of state transition systems. *IEEE Transactions on Visualization and Computer Graphics*, 8(4):319–329, 2002.

[IDS01] IDS Scheer AG. *XML-Export und -Import mit ARIS 5.0, January 2001*, 2001.

[IDS03a] IDS Scheer AG. *ARIS 6 Collaborative Suite Methods Manual*, 2003.

[IDS03b] IDS Scheer AG. *XML-Export und -Import (ARIS 6 Collaborative Suite Version 6.2 Schnittstellenbeschreibung)*. ftp://ftp.ids-scheer.de/pub/ARIS/HELPDESK/EXPORT/, Juni 2003.

[IEE83] IEEE. *IEEE Standard 729: Glossary of Software Engineering Terminology*. IEEE Computer Society Press, 1983.

[IEE90] IEEE. *IEEE Std 610.12-1990 IEEE Standard Glossary of Software Engineering Terminology* . IEEE Computer Society Press, 1990.

[ISO91] International Standards Organisation ISO. Information technology - software product evaluation - quality characteristics and guide lines for their use. Iso/iec is 9126, 1991.

[ISR02] ISR. Editorial Statement and Policy. *Information Systems Research*, 13(4):inside front cover, December 2002.

[JB96] S. Jablonski and C. Bussler. *Workflow Management: Modeling Concepts, Architecture, and Implementation*. International Thomson Computer Press, London, UK, 1996.

[Jes06] J. Jeston. *Business Process Management. Practical Guidelines to Successful Implementations*. Butterworth Heinemann, 2006.

[JHG+88] G.G. Judge, R.C. Hill, W.E. Griffiths, H. Lütkepohl, and T.-C. Lee. *Introduction to the theory and practice of econometrics*. John Wiley & Sons, 2nd edition, 1988.

[JKSK00] S. Junginger, H. Kühn, R. Strobl, and D. Karagiannis. Ein geschäftsprozessmanagement-werkzeug der nächsten generation - adonis: Konzeption und anwendungen. *Wirtschaftsinformatik*, 42(5):392–401, 2000.

[JKW07] K. Jensen, L.M. Kristensen, and L. Wells. Coloured Petri nets and CPN tools for modelling and validation of concurrent systems. *International Journal on Software Tools for Technology Transfer*, 2007.

[Joh95] P. Johannesson. Representation and communication - a speech act based approach to information systems design. *Information Systems*, 20(4):291–303, 1995.

[Jon86] C. Jones. *Programmer Productivity*. McGraw-Hill, 1986.

[JVAR06] M. H. Jansen-Vullers, Wil M. P. van der Aalst, and M. Rosemann. Mining configurable enterprise information systems. *Data & Knowledge Engineering*, 56(3):195–244, 2006.

[Kan02] S.H. Kan. *Metrics and Models in Software Quality Engineering*. Addison Wesley, 2nd edition, 2002.

[KBR+05] N. Kavantzas, D. Burdett, G. Ritzinger, T. Fletcher, Y. Lafon, and C. Barreto. Web Services Choreography Description Language Version 1.0. W3C Candidate Recommendation 9 November 2005, World Wide Web Consortium, April 2005.

[Kha04] R.N. Khan. *Business Process Management: A Practical Guide*. Meghan Kiffer, 2004.

[KHB00] B. Kiepuszewski, A.H.M. ter Hofstede, and C. Bussler. On structured workflow modelling. In B. Wangler and L. Bergman, editors, *Advanced Information Systems Engineering, 12th International Conference CAiSE 2000, Stockholm, Sweden, June 5-9, 2000, Proceedings*, volume 1789 of *Lecture Notes in Computer Science*, pages 431–445. Springer, 2000.

[KHHS93] C. Kruse, A. Hars, R. Heib, and A.-W. Scheer. Ways of utilizing reference models for data engineering in CIM. *International Journal of Flexible Automation and Integrated Manufactoring (FAIM)*, 1(1):47–58, 1993.

[KHP+00] S. Kirn, C. Heine, M. Petsch, F. Puppe, F. Klügl, and R. Herrler. Agentenorientierte Modellierung vernetzter Logistikkreisläufe als Ausgangspunkt agentenbasierter Simulation. In S. Kirn and M. Petsch, editors, *Tagungsband zum 2. Kolloquium des DFG-Schwerpunktprogramms "Intelligente Softwareagenten und Betriebswirtschaftliche Anwendungen"*. TU Ilmenau, Lehrstuhl für Wirtschaftsinformatik 2, 2000.

[Kie03] B. Kiepuszewski. *Expressiveness and Suitability of Languages for Control Flow Modelling in Workflows*. PhD thesis, Queensland University of Technology, Brisbane, Australia, 2003.

[Kin03] E. Kindler. On the semantics of EPCs: A framework for resolving the vicious circle (Extended Abstract). In M. Nüttgens, F.J. Rump, editor, *Proc. of the 2nd GI-Workshop on Business Process Management with Event-Driven Process Chains (EPK 2003), Bamberg, Germany*, pages 7–18, 2003.

[Kin04] E. Kindler. On the semantics of EPCs: A Framework for resolving the vicious circle. In J. Desel and B. Pernici and M. Weske, editor, *Business Process Management, 2nd International Conference, BPM 2004*, volume 3080 of *Lecture Notes in Computer Science*, pages 82–97, 2004.

[Kin06] E. Kindler. On the semantics of EPCs: Resolving the vicious circle. *Data & Knowledge Engineering*, 56(1):23–40, 2006.

[KK02] D. Karagiannis and H. Kühn. Metamodelling Platforms. Invited Paper. In K. Bauknecht and A. Min Tjoa and G. Quirchmayer, editor, *Proceedings of the 3rd International Conference EC-Web 2002 - Dexa 2002, Aix-en-Provence, France*, volume 2455 of *Lecture Notes in Computer Science*, pages 182–196, 2002.

[KLR96] S. Kelly, K. Lyytinen, and M. Rossi. Metaedit+: A fully configurable multi-user and multi-tool case and came environment. In P. Constantopoulos, J. Mylopoulos, and Y. Vassiliou, editors, *Advances Information System Engineering, 8th International Conference, CAiSE'96, Heraklion, Crete, Greece, May 20-24, 1996, Proceedings*, volume 1080 of *Lecture Notes in Computer Science*, pages 1–21. Springer, 1996.

[KLTPZ05] D. Koschützki, K.A. Lehmann, D. Tenfelde-Podehl, and O. Zlotowski. Advanced centrality concepts. In U. Brandes and T. Erlebach, editors, *Network Analysis: Methodological Foundations [outcome of a Dagstuhl seminar, 13-16 April 2004]*, volume 3418 of *Lecture Notes in Computer Science*, pages 83–111. Springer, 2005.

[KN92] R.S. Kaplan and D.P. Norton. The balanced scorecard - measures that drive performance. *Harvard Business Review*, 70(1):71–79, 1992.

[KN00] R.S. Kaplan and D.P. Norton. Having trouble with your strategy? then map it. *Harvard Business Review*, 78(5):167–176, 2000.

[KNS92] G. Keller, M. Nüttgens, and A.-W. Scheer. Semantische Prozessmodellierung auf der Grundlage "Ereignisgesteuerter Prozessketten (EPK)". Heft 89, Institut für Wirtschaftsinformatik, Saarbrücken, Germany, 1992.

[Koh95] R. Kohavi. A study of cross-validation and bootstrap for accuracy estimation and model selection. In *Proceedings of the Fourteenth International Joint Conference on Artificial Intelligence (IJCAI)*, pages 1137–1145, 1995.

[Kos61] E. Kosiol. Modellanalyse als Grundlage unternehmerischer Entscheidungen. *Zeitschrift für betriebswirtschaftlicher Forschung*, pages 318–334, 1961.

[Kos62] E. Kosiol. *Organisation der Unternehmung*. Betriebswirtschaftlicher Verlag Dr. Th. Gabler, 1962.

[KRM06] L.J. Krajewski, L.P. Ritzman, and M.K. Malhotra. *Operations Management. Processes and Value Chains: Process and Value Chains*. Addison-Wesley, 2006.

[KRS05] E. Kindler, V. Rubin, and W. Schäfer. Incremental Workflow mining based on Document Versioning Information. In M. Li, B. Boehm, and L.J. Osterweil, editors, *Proc. of the Software Process Workshop 2005, Beijing, China*, volume 3840 of *LNCS*, pages 287–301. SPRINGER, May 2005.

[KRS06a] E. Kindler, V. Rubin, and W. Schäfer. Activity mining for discovering software process models. In B. Biel, M. Book, and V. Gruhn, editors, *Proc. of the Software Engineering 2006 Conference, Leipzig, Germany*, volume P-79 of *LNI*, pages 175–180. Gesellschaft für Informatik, March 2006.

[KRS06b] E. Kindler, V. Rubin, and W. Schäfer. Process Mining and Petri Net Synthesis. In J. Eder and S. Dustdar, editors, *Business Process Management Workshops*,

volume 4103 of *Lecture Notes in Computer Science (LNCS)*, pages 105–116. Springer Verlag, September 2006.

[Kru96] C. Kruse. *Referenzmodellgestütztes Geschäftsprozeßmanagement: Ein Ansatz zur prozeßorientierten Gestaltung betriebslogistischer Systeme*. Gabler Verlag, 1996.

[KSJ06] J. Krogstie, G. Sindre, and H.D. Jørgensen. Process models representing knowledge for action: a revised quality framework. *European Journal of Information Systems*, 15(1):91–102, 2006.

[KT98] G. Keller and T. Teufel. *SAP(R) R/3 Process Oriented Implementation: Iterative Process Prototyping*. Addison-Wesley, 1998.

[Kug02] M. Kugeler. *Prozessmanagement*, chapter SCM und CRM - Prozessmodellierung für Extende Enterprises, pages 457–485. Springer-Verlag, 3rd edition, 2002.

[Küh06] T. Kühne. Matters of (meta-) modeling. *Software and Systems Modeling*, 5(4):369–385, 2006.

[KUL06] O. Kopp, T. Unger, and F. Leymann. Nautilus Event-driven Process Chains: Syntax, Semantics, and their mapping to BPEL. In M. Nüttgens and F.J. Rump and J. Mendling, editor, *Proceedings of the 5th GI Workshop on Business Process Management with Event-Driven Process Chains (EPK 2006)*, pages 85–104, Vienna, Austria, December 2006. German Informatics Society.

[LA94] F. Leymann and W. Altenhuber. Managing business processes as an information resource. *IBM Systems Journal*, 33(2):326–348, 1994.

[LA98] P. Loos and T. Allweyer. Process Orientation and Object-Orientation An Approach for Integrating UML and Event-Driven Process Chains (EPC). Heft 144, Institut für Wirtschaftsinformatik, Saarbrücken, Germany, 1998.

[LB06] L.M. Laird and M.C. Brennan. *Software Measurement and Estimation: A Practical Approach*. IEEE Computer Society and John Wiley & Sons, Inc., Hoboken, New Jersey, 2006.

[Leh97] F. Lehner. *Lexikon der Wirtschaftsinformatik*, chapter Wirtschaftsinformatik, Forschungsgegenstände und Erkenntisverfahren, pages 438–439. Springer-Verlag, 3rd edition, 1997.

[LG06] R. Laue and V. Gruhn. Complexity metrics for business process models. In Witold Abramowicz and Heinrich C. Mayr, editors, *9th International Conference on Business Information Systems (BIS 2006)*, volume 85 of *Lecture Notes in Informatics*, pages 1–12, 2006.

[LK01] A.M. Latva-Koivisto. Finding a complexity for business process models. Research report, Helsinki University of Technology, February 2001.

[LK05] R. Liu and A. Kumar. An analysis and taxonomy of unstructured workflows. In W.M.P. van der Aalst, B. Benatallah, F. Casati, and F. Curbera, editors, *Business Process Management, 3rd International Conference, BPM 2005, Nancy, France, September 5-8, 2005, Proceedings*, volume 3649 of *Lecture Notes in Computer Science*, pages 268–284, 2005.

[LK06] B. List and B. Korherr. An evaluation of conceptual business process modelling languages. In H. Haddad, editor, *Proceedings of the 2006 ACM Symposium on Applied Computing (SAC), Dijon, France, April 23-27, 2006*, pages 1532–1539. ACM, 2006.

[LL00] M. Leuschel and H. Lehmann. Coverability of reset petri nets and other well-structured transition systems by partial deduction. In J.W. Lloyd, V. Dahl, U. Furbach, M. Kerber, K.-K. Lau, C. Palamidessi, L.M. Pereira, Y. Sagiv, and P.J. Stuckey, editors, *Computational Logic - CL 2000, First International Conference, London, UK, 24-28 July, 2000, Proceedings*, volume 1861 of *Lecture Notes in Computer Science*, pages 101–115. Springer, 2000.

[LLS06] K.C. Laudon, J.P. Laudon, and D. Schoder. *Wirtschaftsinformatik. Eine Einführung*. Pearson Studium, 2006.

[LM04] M. Laguna and J. Marklund. *Business Process Modeling, Simulation and Design*. Prentice Hall, 2004.

[Löw00] J. Löwer. *tDOM A fast XML/DOM/XPath package for Tcl written in C*. http://www.tdom.org/documents/tDOM3.pdf, June 2000. Document to Talk at the First European Tcl/Tk User Meeting.

[LR00] F. Leymann and D. Roller. *Production Workflow - Concepts and Techniques*. Prentice Hall, 2000.

[LS02] K.R. Lang and M. Schmidt. Workflow-supported organizational memory systems: An industrial application. In *35th Hawaii International International Conference on Systems Science (HICSS-35 2002), CD-ROM / Abstracts Proceedings*, page 208. IEEE Computer Society, 2002.

[LSS94] Odd Ivar Lindland, Guttorm Sindre, and Arne Sølvberg. Understanding quality in conceptual modeling. *IEEE Software*, 11(2):42–49, 1994.

[LSW97a] P. Langner, C. Schneider, and J. Wehler. Ereignisgesteuerte Prozeßketten und Petri-Netze. Report Series of the Department of Computer Science 196, University of Hamburg - Computer Sience Department, March 1997. in German.

[LSW97b] P. Langner, C. Schneider, and J. Wehler. Prozeßmodellierung mit ereignisgesteuerten Prozeßketten (EPKs) und Petri-Netzen. *Wirtschaftsinformatik*, 39(5):479–489, 1997.

[LSW98] P. Langner, C. Schneider, and J. Wehler. Petri Net Based Certification of Event driven Process Chains. In J. Desel and M. Silva, editor, *Application and Theory of Petri Nets*, volume 1420 of *Lecture Notes in Computer Science*, pages 286–305, 1998.

[LY90] G.S. Lee and J.-M. Yoon. An empirical study on the complexity metrics of petri nets. In *JTC-CSCC: Joint Technical Conference on Circuits Systems, Computers and Communications, 1990*, pages 327–332, 1990.

[LY92] G.S. Lee and J.-M. Yoon. An empirical study on the complexity metrics of petri nets. *Microelectronics and Reliability*, 32(3):323–329, 1992.

[LZLC02] H. Lin, Z. Zhao, H. Li, and Z. Chen. A novel graph reduction algorithm to identify structural conflicts. In *35th Hawaii International Conference on System Sciences (HICSS-35 2002), CD-ROM / Abstracts Proceedings, 7-10 January 2002, Big Island, HI, USA. Track 9*. IEEE Computer Society, 2002.

[MA06] J. Mendling and W.M.P. van der Aalst. Towards EPC Semantics based on State and Context. In M. Nüttgens and F.J. Rump and J. Mendling, editor, *Proceedings of the 5th GI Workshop on Business Process Management with Event-Driven Process Chains (EPK 2006)*, pages 25–48, Vienna, Austria, December 2006. German Informatics Society.

[MADV06] J. Mendling, W.M.P. van der Aalst, B.F. van Dongen, and H.M.W. Verbeek. Referenzmodell: Sand im Getriebe - Webfehler. *iX - Magazin für Professionelle Informationstechnik. (in German)*, pages 131–133, August 2006.

[Mar03] A. Martens. On Compatibility of Web Services. *Petri Net Newsletter*, 65:12–20, 2003.

[MB89] T.J. McCabe and C.W. Butler. Design complexity measurement and testing. *Communications of the ACM*, 32:1415–1425, 1989.

[MBK+98] P. Mertens, F. Bodendorf, W. König, A. Picot, and M. Schumann. *Grundzüge der Wirtschaftsinformatik*. Springer-Verlag, 5th edition, 1998.

[McC76] T.J. McCabe. A complexity measure. *IEEE Transactions on Software Engineering*, 2(4):308–320, 1976.

[MCH03] T.W. Malone, K. Crowston, and G.A. Herman, editors. *Organizing Business Knowledge: The Mit Process Handbook: The MIT Process Handbook*. The MIT Press, 2003.

[MDF05] G. Marczyk, D. DeMatteo, and D. Festinger. *Essentials of Research Design and Methodology*. John Wiley & Sons, Inc., 2005.

[MH04] D.L. McGuinness and F. van Harmelen. OWL Web Ontology Language Overview. W3c recommendation, World Wide Web Consortium, 2004.

[MH05] J. Mendling and M. Hafner. From Inter-Organizational Workflows to Process Execution: Generating BPEL from WS-CDL. In Robert Meersman, Zahir Tari, and Pilar Herrero, editors, *Proceedings of OTM 2005 Workshops*, volume 3762 of *Lecture Notes in Computer Science*, 2005.

[MLZ05] J. Mendling, K.B. Lassen, and U. Zdun. Transformation strategies between block-oriented and graph-oriented process modelling languages. Technical Report JM-2005-10-10, WU Vienna, October 2005.

[MLZ06a] J. Mendling, K.B. Lassen, and U. Zdun. Experiences in enhancing existing BPM Tools with BPEL Import and Export. In J.L. Fiadeiro S. Dustdar and A. Sheth, editors, *Proceedings of BPM 2006*, volume 4102 of *Lecture Notes in Computer Science*, pages 348–357, Vienna, Austria, 2006. Springer-Verlag.

[MLZ06b] J. Mendling, K.B. Lassen, and U. Zdun. Transformation strategies between block-oriented and graph-oriented process modelling languages. In F. Lehner, H. Nösekabel, and P. Kleinschmidt, editors, *Multikonferenz Wirtschaftsinformatik 2006, XML4BPM Track, Band 2*, pages 297–312, Passau, Germany, February 2006. GITO-Verlag Berlin.

[MMG02] M.L. Markus, A. Majchrzak, and L. Gasser. A Design Theory for Systems that Support Emergent Knowledge Processes. *MIS Quarterly*, 26(3):554–578, September 2002.

[MMN06a] J. Mendling, M. Moser, and G. Neumann. Transformation of yEPC Business Process Models to YAWL. In *Proceedings of the 21st Annual ACM Symposium on Applied Computing*, volume 2, pages 1262–1267, Dijon, France, 2006. ACM.

[MMN+06b] J. Mendling, M. Moser, G. Neumann, H.M.W. Verbeek, B.F. van Dongen, and W.M.P. van der Aalst. A Quantitative Analysis of Faulty EPCs in the SAP Reference Model. BPM Center Report BPM-06-08, BPMCenter.org, 2006.

[MMN+06c] J. Mendling, M. Moser, G. Neumann, H.M.W. Verbeek, B.F. van Dongen, and W.M.P. van der Aalst. Faulty EPCs in the SAP Reference Model. In

J.L. Fiadeiro S. Dustdar and A. Sheth, editors, *Proceedings of BPM 2006*, volume 4102 of *Lecture Notes in Computer Science*, page 451457, Vienna, Austria, 2006. Springer-Verlag.

[MMP05] J. Mendling, M. zur Muehlen, and A. Price. *Process Aware Information Systems: Bridging People and Software Through Process Technology*, chapter Standards for Workflow Definition and Execution, pages 281–316. Wiley Publishing, September 2005.

[MN02] J. Mendling and M. Nüttgens. Event-Driven-Process-Chain-Markup-Language (EPML): Anforderungen zur Definition eines XML-Schemas für Ereignisgesteuerte Prozessketten (EPK). In M. Nüttgens and F.J. Rump, editor, *Proc. of the 1st GI-Workshop on Business Process Management with Event-Driven Process Chains (EPK 2002), Trier, Germany*, pages 87–93, 2002.

[MN03a] J. Mendling and M. Nüttgens. EPC Modelling based on Implicit Arc Types. In M. Godlevsky and S. W. Liddle and H. C. Mayr, editor, *Proc. of the 2nd International Conference on Information Systems Technology and its Applications (ISTA), Kharkiv, Ukraine*, volume 30 of *Lecture Notes in Informatics*, pages 131–142, 2003.

[MN03b] J. Mendling and M. Nüttgens. EPC Syntax Validation with XML Schema Languages. In M. Nüttgens and F.J. Rump, editor, *Proc. of the 2nd GI-Workshop on Business Process Management with Event-Driven Process Chains (EPK 2003), Bamberg, Germany*, pages 19–30, 2003.

[MN03c] J. Mendling and M. Nüttgens. XML-basierte Geschäftsprozessmodellierung. In W. Uhr and W. Esswein and E. Schoop, editor, *Proc. of Wirtschaftsinformatik 2003 / Band II, Dresden, Germany*, pages 161 –180, 2003.

[MN04a] J. Mendling and M. Nüttgens. Exchanging EPC Business Process Models with EPML. In M. Nüttgens and J. Mendling, editors, *XML4BPM 2004, Proceedings of the 1st GI Workshop XML4BPM – XML Interchange Formats for Business Process Management at 7th GI Conference Modellierung 2004, Marburg Ger-*

many, pages 61–80, http://wi.wu-wien.ac.at/~mendling/XML4BPM/xml4bpm-2004-proceedings-epml.pdf, March 2004.

[MN04b] J. Mendling and M. Nüttgens. Transformation of ARIS Markup Language to EPML. In M. Nüttgens and F.J. Rump, editor, *Proceedings of the 3rd GI Workshop on Business Process Management with Event-Driven Process Chains (EPK 2004)*, pages 27–38, 2004.

[MN04c] J. Mendling and M. Nüttgens. XML-based Reference Modelling: Foundations of an EPC Markup Language. In J. Becker, editor, *Referenzmodellierung - Proceedings of the 8th GI-Workshop on Reference Modelling, MKWI Essen, Germany*, pages 51–71, 2004.

[MN05] J. Mendling and M. Nüttgens. EPC Markup Language (EPML) - An XML-Based Interchange Format for Event-Driven Process Chains (EPC). Technical Report JM-2005-03-10, Vienna University of Economics and Business Administration, Austria, 2005.

[MN06] J. Mendling and M. Nüttgens. EPC Markup Language (EPML) - An XML-Based Interchange Format for Event-Driven Process Chains (EPC). *Information Systems and e-Business Management*, 4(3):245 – 263, 2006.

[MNN04] J. Mendling, M. Nüttgens, and G. Neumann. A Comparison of XML Interchange Formats for Business Process Modelling. In F. Feltz, A. Oberweis, and B. Otjacques, editors, *Proceedings of EMISA 2004 - Information Systems in E-Business and E-Government*, volume 56 of *Lecture Notes in Informatics*, 2004.

[MNN05a] J. Mendling, G. Neumann, and M. Nüttgens. Towards Workflow Pattern Support of Event-Driven Process Chains (EPC). In M. Nüttgens and J. Mendling, editors, *XML4BPM 2005, Proceedings of the 2nd GI Workshop XML4BPM – XML for Business Process Management at the 11th GI Conference BTW 2005*, volume 145 of *CEUR Workshop Proceedings*, pages 23–38, Karlsruhe, Germany, March 2005. Sun SITE Central Europe.

[MNN05b] J. Mendling, G. Neumann, and M. Nüttgens. *Workflow Handbook 2005*, chapter A Comparison of XML Interchange Formats for Business Process Modelling, pages 185–198. Future Strategies Inc., Lighthouse Point, FL, USA, 2005.

[MNN05c] J. Mendling, G. Neumann, and M. Nüttgens. Yet Another Event-Driven Process Chain. In *Proceedings of BPM 2005*, volume 3649 of *Lecture Notes in Computer Science*, 2005.

[MNN05d] J. Mendling, G. Neumann, and M. Nüttgens. Yet Another Event-driven Process Chain - Modeling Workflow Patterns with yEPCs. *Enterprise Modelling and Information Systems Architectures - an International Journal*, 1(1):3–13, October 2005.

[Moo01] D.L. Moody. *Dealing With Complexity: A Practical Method For Representing Large Entity Relationship Models*. PhD thesis, Department of Information Systems, University of Melbourne, June 2001.

[Moo03] D.L. Moody. Measuring the quality of data models: an empirical evaluation of the use of quality metrics in practice. In *Proceedings of the 11th European Conference on Information Systems, ECIS 2003, Naples, Italy 16-21 June 2003*, 2003.

[Moo05] D.L. Moody. Theoretical and practical issues in evaluating the quality of conceptual models: current state and future directions. *Data & Knowledge Engineering*, 55(3):243–276, 2005.

[Mor99] S. Morasca. Measuring attributes of concurrent software specifications in petri nets. In *METRICS '99: Proceedings of the 6th International Symposium on Software Metrics*, pages 100–110, Washington, DC, USA, 1999. IEEE Computer Society.

[MR] J. Mendling and J. Recker. Extending the discussion of model quality: Why clarity and completeness may not be enough. In *Proceedings of the CAiSE*

Workshops at the 19th Conference on Advanced Information Systems Engineering (CAiSE 2007), YEAR =.

[MR04] M. zur Muehlen and M. Rosemann. Multi-Paradigm Process Management. In *Proc. of the Fifth Workshop on Business Process Modeling, Development, and Support - CAiSE Workshops*, 2004.

[MRRA06] J. Mendling, J. Recker, M. Rosemann, and Wil M.P. van der Aalst. Generating Correct EPCs from Configured CEPCs. In *Proceedings of the 21st Annual ACM Symposium on Applied Computing*, volume 2, pages 1505–1511, Dijon, France, 2006. ACM.

[MS95] S.T. March and G. Smith. Design and Natural Science Research on Information Technology. *Decision Support Systems*, 15(4):251–266, December 1995.

[MS06] J. Mendling and C. Simon. Business Process Design by View Integration. In Johann Eder and Schahram Dustdar, editors, *Proceedings of BPM Workshops 2006*, volume 4103 of *Lecture Notes in Computer Science*, pages 55–64, Vienna, Austria, 2006. Springer-Verlag.

[MSBS02] D.L. Moody, G. Sindre, T. Brasethvik, and Arne Sølvberg. Evaluating the quality of process models: Empirical testing of a quality framework. In Stefano Spaccapietra, Salvatore T. March, and Yahiko Kambayashi, editors, *Conceptual Modeling - ER 2002, 21st International Conference on Conceptual Modeling, Tampere, Finland, October 7-11, 2002, Proceedings*, volume 2503 of *Lecture Notes in Computer Science*, pages 380–396. Springer, 2002.

[MT77] F. Mosteller and J. W. Tukey. *Data Analysis and Regression*. Addison-Wesley, 1977.

[Mue04] M. zur Muehlen. *Workflow-based Process Controlling. Foundation, Design, and Implementation of Workflow-driven Process Information Systems.*, volume 6 of *Advances in Information Systems and Management Science*. Logos, Berlin, 2004.

[Mur89] T. Murata. Petri Nets: Properties, Analysis and Applications. *Proceedings of the IEEE*, 77(4):541–580, April 1989.

[MW94] T.J. McCabe and A.H. Watson. Software complexity. *Journal of Defence Software Engineering*, 7(12):5–9, 1994. Crosstalk.

[MZ05a] J. Mendling and J. Ziemann. Transformation of BPEL Processes to EPCs. In M. Nüttgens and F.J. Rump, editor, *Proceedings of the 4th GI Workshop on Business Process Management with Event-Driven Process Chains (EPK 2005)*, pages 41–53, Hamburg, Germany, December 2005. German Informatics Society.

[MZ05b] J. Mendling and J. Ziemann. Transformation of BPEL Processes to EPCs. In *4th GI Workshop on Event-Driven Process Chains*, Hamburg, Germany, 2005. German Informatics Society.

[Nag91] N.J.D. Nagelkerke. A note on a general definition of the coefficient of determination. *Biometrika*, 78(3):691–692, September 1991.

[Neu88] G. Neumann. *Metaprogrammierung und Prolog*. Addison-Wesley, December 1988.

[NFM00] N. Fridman Noy, R.W. Fergerson, and M.A. Musen. The knowledge model of protégé-2000: Combining interoperability and flexibility. In R. Dieng and O. Corby, editors, *Knowledge Acquisition, Modeling and Management, 12th International Conference, EKAW 2000, Juan-les-Pins, France, October 2-6, 2000, Proceedings*, volume 1937 of *Lecture Notes in Computer Science*, pages 17–32. Springer, 2000.

[Nis94] M.E. Nissen. Valuing it through virtual process measurement. In *Proc. 15th. International Conference on Information Systems, Vancouver, Canada*, pages 309–323, 1994.

[Nis96] M.E. Nissen. *Knowledge-based organizational process redesign: using process flow measures to transform procurement*. PhD thesis, University of South California, 1996.

[Nis98] M.E. Nissen. Redesigning reengineering through measurement-driven inference. *MIS Quarterly*, 22(4):509–534, 1998.

[Nor32] F. Nordsieck. *Die Schaubildliche Erfassung und Untersuchung der Betriebsorganisation*. Organisation - Eine Schriftenreihe. C. E. Poeschel Verlag, Stuttgart, 1932.

[Nor34] F. Nordsieck. *Grundlagen der Organisationslehre*. C. E. Poeschel Verlag, 1934.

[NPW03] S. Neumann, C. Probst, and C. Wernsmann. Continuous Process Management. In J. Becker, M. Kugeler, and M. Rosemann, editors, *Process Management: A Guide for the Design of Business Processes*, pages 233–250. Springer, Berlin, New York, 2003.

[NR02] M. Nüttgens and F.J. Rump. Syntax und Semantik Ereignisgesteuerter Prozessketten (EPK). In J. Desel and M. Weske, editor, *Proceedings of Promise 2002, Potsdam, Germany*, volume 21 of *Lecture Notes in Informatics*, pages 64–77, 2002.

[Nüt95] M. Nüttgens. *Koordiniert-dezentrales Informationsmanagement: Rahmenkonzept, Koordinationsmodelle und Werkzeug-Shell*. PhD thesis, Rechts- und Wirtschaftswissenschaftliche Fakultät der Universität des Saarlandes, 1995.

[NZ00] G. Neumann and U. Zdun. XOTcl, an Object-Oriented Scripting Language. In *Proc. of the 7th USENIX Tcl/Tk Conference, Austin, Texas, USA*, 2000.

[Obe96] A. Oberweis. An integrated approach for the specification of processes and related complex structured objects in business applications. *Decision Support Systems*, 17:31–53, 1996.

[ÖG92] H. Österle and T. Gutzwiller. *Konzepte angewandter Analyse- und Design-Methoden. Band 1: Ein Referenz-Metamodell für die Analyse und das System-Design*. Angewandte InformationsTechnik-Verl. GmbH, 1992.

[OMG02] OMG, ed. Meta Object Facility. Version 1.4, Object Management Group, 2002.

[OMG04] OMG, ed. Unified Modeling Language. Version 2.0, Object Management Group, 2004.

[OMG06] OMG, ed. Business Process Modeling Notation (BPMN) Specification. Final Adopted Specification, dtc/06-02-01, Object Management Group, February 2006.

[OS96] A. Oberweis and P. Sander. Information system behavior specification by high-level petri nets. *ACM Transactions on Information Systems*, 14(4):380–420, 1996.

[Öst95] H. Österle. *Business Engineering Prozeß- und Systementwicklung. Band 1: Entwurfstechniken*. Springer Verlag, 1995.

[Pal07] Nathaniel Palmer. A survey of business process initiatives. BPT Report, Business Process Trends and Transformation+Innovation, January 2007.

[Pem02] S. Pemberton et al. XHTML 1.0 The Extensible HyperText Markup Language (Second Edition). W3C Recommendation 26 January 2000, revised 1 August 2002, World Wide Web Consortium, 2002.

[Pet62] C.A. Petri. *Kommunikation mit Automaten*. PhD thesis, Fakultät für Mathematik und Physik, Technische Hochschule Darmstadt, Darmstadt, Germany, 1962.

[PH07] S. Philippi and H.J. Hill. Communication support for systems engineering - process modelling and animation with april. *The Journal of Systems and Software,*, 2007. accepted for publication.

[Pnu77] A. Pnueli. The Temporal Logic of Programs. In *Proceedings of the 18th IEEE Annual Symposium on the Foundations of Computer Science*, pages 46–57. IEEE Computer Society Press, Providence, 1977.

[Por85] M. E. Porter. *Competitive Advantage : Creating and Sustaining Superior Performance*. The Free Press, New York, 1985.

[Pri95] J. Priemer. *Entscheidungen über die Einsetzbarkeit von Software anhand formaler Modelle*. PhD thesis, Westfälische Wilhelms-Universität Münster, 1995.

[PW05] A.J. Pretorius and J.J. van Wijk. Multidimensional visualization of transition systems. In *9th International Conference on Information Visualisation, IV 2005, 6-8 July 2005, London, UK*, pages 323–328. IEEE Computer Society, 2005.

[PW06a] A.J. Pretorius and J.J. van Wijk. Visual analysis of multivariate state transition graphs. *IEEE Visualization and Computer Graphics*, 12(5):685–692, 2006.

[PW06b] F. Puhlmann and M. Weske. Investigations on soundness regarding lazy activities. In S. Dustdar, J.L. Fiadeiro, and A. Sheth, editors, *Business Process Management, 4th International Conference, BPM 2006*, volume 4102 of *Lecture Notes in Computer Science*, pages 145–160. Springer-Verlag, September 2006.

[RA07] M. Rosemann and Wil van der Aalst. A Configurable Reference Modelling Language. *Information Systems*, 32:1–23, 2007.

[RD98] M. Reichert and P. Dadam. ADEPTflex: Supporting Dynamic Changes of Workflow without Loosing Control. *Journal of Intelligent Information Systems*, 10(2):93–129, 1998.

[Rei03] H.A. Reijers. A cohesion metric for the definition of activities in a workflow process. In *Proceedings of the Eighth CAiSE/IFIP8.1 International Workshop on Evaluation of Modeling Methods in Systems Analysis and Design (EMMSAD 2003)*, pages 116–125, 2003.

[RG02] M. Rosemann and P. Green. Developing a meta model for the Bunge-Wand-Weber ontological constructs. *Information Systems*, 27:75–91, 2002.

[RGA+06] V. Rubin, C.W. Günther, W.M.P. van der Aalst, E. Kindler, B.F. van Dongen, and W. Schäfer. Process mining framework for software processes. BPMCenter Report BPM-07-01, BPMcenter.org, 2006.

[Rit99] P. Rittgen. Modified EPCs and Their Formal Semantics. Arbeitsberichte des Instituts für Wirtschaftsinformatik 19, Universität Koblenz-Landau, 1999.

[Rit00] P. Rittgen. Paving the Road to Business Process Automation. In *Proc. of the European Conference on Information Systems (ECIS), Vienna, Austria*, pages 313–319, 2000.

[RMRA06] J. Recker, J. Mendling, M. Rosemann, and W.M.P. van der Aalst. Model-driven Enterprise Systems Configuration. In *Proceedings of the 18th Conference on Advanced Information Systems Engineering (CAiSE 2006)*, volume 4001 of *Lecture Notes in Computer Science*, pages 369–383, Luxembourg, Luxembourg, 2006. Springer-Verlag.

[Rod02] J. Rodenhagen. Ereignisgesteuerte Prozessketten - Multi-Instantiierungsfähigkeit und referentielle Persistenz. In *Proceedings of the 1st GI Workshop on Business Process Management with Event-Driven Process Chains*, pages 95–107, 2002.

[Ros95] M. Rosemann. *Erstellung und Integration von Prozeßmodellen - Methodenspezifische Gestaltungsempfehlungen für die Informationsmodellierung*. PhD thesis, Westfälische Wilhelms-Universität Münster, 1995.

[Ros03] M. Rosemann. *Process Management: A Guide for the Design of Business Processes*, chapter Preparation of Process Modeling, pages 41–78. Springer-Verlag, 2003.

[Ros06] M. Rosemann. Potential pitfalls of process modeling: part a. *Business Process Management Journal*, 12(2):249–254, 2006.

[Rum99] F.J. Rump. *Geschäftsprozessmanagement auf der Basis ereignisgesteuerter Prozessketten - Formalisierung, Analyse und Ausführung von EPKs*. Teubner Verlag, 1999.

[RV04] H.A. Reijers and I.T.P. Vanderfeesten. Cohesion and coupling metrics for workflow process design. In J. Desel, B. Pernici, and M. Weske, editors, *Business Process Management: Second International Conference, BPM 2004, Potsdam, Germany, June 17-18, 2004. Proceedings*, volume 3080 of *Lecture Notes in Computer Science*, pages 290–305. Springer, 2004.

[SA05] S.M. Smith and G.S. Albaum. *Fundamentals of Marketing Research*. Sage Publications, 2005.

[SAJ+02] E. Söderström, B. Andersson, P. Johannesson, E. Perjons, and B. Wangler. Towards a Framework for Comparing Process Modelling Languages. In A. Banks Pidduck, J. Mylopoulos, C.C. Woo, and M.T. Özsu, editors, *Proceedings of the 14th International Conference on Advanced Information Systems Engineering (CAiSE)*, volume 2348 of *Lecture Notes in Computer Science*, pages 600–611, 2002.

[SB91] R.W. Selby and V.R. Basili. Analyzing error-prone system structure. *IEEE Transactions on Software Engineering*, 17(2):141–152, 1991.

[Sch76] A.-W. Scheer. *Produktionsplannung auf der Grundlage einer Datenbank des Fertigungsbereichs*. Oldenbourg, 1976.

[Sch78] A.-W. Scheer. *Wirtschafts- und Betriebsinformatik*. Verlag Moderne Industrie, München, 1978.

[Sch84] A.-W. Scheer. *EDV-orientierte Betriebswirtschaftslehre*. Springer-Verlag, Berlin, Heidelberg, New York, Tokio, 1984.

[Sch87] A.-W. Scheer. *CIM - Computer integrated manufacturing : der computergesteuerte Industriebetrieb*. Springer-Verlag, 1987.

[Sch98a] A.-W. Scheer. *ARIS - Business Process Frameworks*. Springer, Berlin et al., 2nd edition, 1998.

[Sch98b] A.-W. Scheer. *Wirtschaftsinformatik: Referenzmodelle für industrielle Geschäftsprozesse*. Springer-Verlag, 2nd edition edition, 1998.

[Sch98c] R. Schütte. *Grundsätze ordnungsgemäßer Referenzmodellierung*. Gabler Verlag, 1998.

[Sch00] A.-W. Scheer. *ARIS - Business Process Modeling*. Springer, Berlin et al., 3rd edition, 2000.

[Sch02] A.-W. Scheer. *ARIS in der Praxis - Gestaltung, Implementierung und Optimierung von Geschäftsprozessen*, chapter ARIS: Von der Vision zur praktischen Geschäftsprozesssteuerung, pages 1–14. Springer-Verlag, 2002.

[SCJ06] N. Slack, S. Chambers, and R. Johnston. *Operations and Process Management. Principles and Practice for Strategic Impact*. Prentice Hall, 2006.

[Sco00] J. Scott. *Social Network Analysis: A Handbook*. SAGE Publications Ltd, 2nd edition, 2000.

[SD95] R. Striemer and W. Deiters. Workflow Management - Erfolgreiche Planung und Durchführung von Workflow-Projekten. Arbeitsbericht, Fraunhofer Institut für Software- und Systemtechnik, 1995.

[SDL05] K. Sarshar, P. Dominitzki, and P. Loos. Einsatz von Ereignisgesteuerten Prozessketten zur Modellierung von Prozessen in der Krankenhausdomäne Eine empirische Methodenevaluation. In M. Nüttgens and F.J. Rump, editor, *Proceedings of the 4th GI Workshop on Business Process Management with Event-Driven Process Chains (EPK 2005)*, pages 97–116, Hamburg, Germany, December 2005. German Informatics Society.

[Sea69] J. Searle. *Speech Acts: An Essay in the Philosophy of Language*. Cambridge University Press, Cambridge, Eng., 1969.

[SEI92] Software Engineering Institute SEI. Software size measurement: A framework for counting source statements. Technical Report CMU/SEI-92-TR-020, ADA258304, 1992.

[Sei02] H. Seidlmeier. *Prozessmodellierung mit ARIS*. Vieweg Verlag, 2002.

[SF06] H. Smith and P. Fingar. *Business Process Management: The Third Wave*. Meghan Kiffer, 2006.

[SH05] P. Stahlknecht and U. Hasenkamp. *Einführung in die Wirtschaftsinformatik*. Springer-Verlag, 11th edition, 2005.

[Sha88] S.M. Shatz. Towards complexity metrics for ada tasking. *IEEE Transaction on Software Engineering*, 14(8):1122–1127, 1988.

[Sil01a] L. Silverston. *The Data Model Resource Book, Volume 1, A Library of Universal Data Models for all Enterprises*. John Wiley and Sons, New York, revised edition, 2001.

[Sil01b] L. Silverston. *The Data Model Resource Book, Volume 2, A Library of Data Models for Specific Industries*. John Wiley and Sons, New York, revised edition, 2001.

[Sim96] H.A. Simon. *Sciences of the Artificial*. The MIT Press, 3rd edition, 1996.

[Sim06] Carlo Simon. *Negotiation Processes. The semantic process language and applications*. University of Koblenz, September 2006. Habilitationsschrift.

[SL05] K. Sarshar and P. Loos. Comparing the control-flow of epc and petri net from the end-user perspective. In W.M.P. van der Aalst, B. Benatallah, F. Casati, and F. Curbera, editors, *Business Process Management, 3rd International Conference, BPM 2005, Nancy, France, September 5-8, 2005, Proceedings*, LNCS 3649, pages 434–439, 2005.

[SLTM91] K. Smolander, K. Lyytinen, V.-P. Tahvanainen, and P. Marttiin. Metaedit - a flexible graphical environment for methodology modelling. In R. Andersen, J.A. Bubenko Jr., and A. Sølvberg, editors, *Advanced Information Systems Engineering, CAiSE'91, Trondheim, Norway, May 13-15, 1991, Proceedings*, volume 498 of *Lecture Notes in Computer Science*, pages 168–193. Springer, 1991.

[SM06] C. Simon and J. Mendling. Verification of Forbidden Behavior in EPCs. In H.C. Mayr and R. Breu, editors, *Proceedings of the GI Conference Modellierung*, volume 82 of *Lecture Notes in Informatics*, pages 233–242, Innsbruck, Austria, March 2006. German Informatics Society.

[Smi76] A. Smith. *An inquiry into the nature and causes of the wealth of nations*. London, 1776.

[Smi07] R.F. Smith. *Business Process Management and the Balanced Scorecard. Focusing Processes as Strategic Drivers: Using Processes as Strategic Drivers*. Wiley & Sons, 2007.

[SNZ97] A.-W. Scheer, M. Nüttgens, and V. Zimmermann. Objektorientierte Ereignisgesteuerte Prozeßketten (oEPK) - Methode und Anwendung. Heft 141, Institut für Wirtschaftsinformatik, Saarbrücken, Germany, 1997.

[SO96] W. Sadiq and M.E. Orlowska. Modeling and verification of workflow graphs. Technical Report No. 386, Department of Computer Science, The University of Queensland, Australia, 1996.

[SO99] W. Sadiq and M.E. Orlowska. Applying graph reduction techniques for identifying structural conflicts in process models. In M. Jarke and A. Oberweis, editors, *Advanced Information Systems Engineering, 11th International Conference CAiSE'99, Heidelberg, Germany, June 14-18, 1999, Proceedings*, volume 1626 of *Lecture Notes in Computer Science*, pages 195–209. Springer, 1999.

[SO00] W. Sadiq and M.E. Orlowska. Analyzing Process Models using Graph Reduction Techniques. *Information Systems*, 25(2):117–134, 2000.

[Som01] I. Sommerville. *Software Engineering*. Addison-Wesley, 6th edition, 2001.

[SR98] R. Schütte and T. Rotthowe. The Guidelines of Modeling - An Approach to Enhance the Quality in Information Models. In Tok Wang Ling, Sudha Ram, and Mong-Li Lee, editors, *Proceedings of the 17th International Conference*

on Conceptual Modeling, volume 1507 of *Lecture Notes in Computer Science*, pages 240–254, Singapore, 1998. Springer.

[SRS96] B. Scholz-Reiter and E. Stickel, editors. *Business Process Modelling*. Springer-Verlag, 1996.

[ST03] K. Schneider and O. Thomas. Kundenorientierte Dienstleistungsmodellierung mit Ereignisgesteuerten Prozessketten. In M. Nüttgens and F.J. Rump, editor, *Proc. of the 2nd GI-Workshop on Business Process Management with Event-Driven Process Chains (EPK 2003), Bamberg, Germany*, pages 87–93, 2003.

[Sta73] H. Stachowiak. *Allgemeine Modelltheorie*. Springer-Verlag, 1973.

[STA05] A.-W. Scheer, O. Thomas, and O. Adam. *Process Aware Information Systems: Bridging People and Software Through Process Technology*, chapter Process Modeling Using Event-Driven Process Chains, pages 119–146. Wiley Publishing, 2005.

[Sta06] J.L. Staud. *Geschäftsprozessanalyse: Ereignisgesteuerte Prozessketten und Objektorientierte Geschäftsprozessmodellierung für Betriebswirtschaftliche Standardsoftware*. Springer-Verlag, 3rd edition, July 2006.

[Ste46] S.S. Stevens. On the theory of scale types and measurement. *Science*, 103:677–680, 1946.

[Ste51] S.S. Stevens. *Handbook of Experimental Psychology*, chapter Mathematics, Measurement, and Psychophysics. John Wiley, 1951.

[Sto74] M. Stone. Cross validity choice and assessment of statistical predictors. *Journal of the Royal Statistical Society*, 36:111–147, 1974.

[Str96] S. Strahringer. *Metamodellierung als Instrument des Methodenvergleichs. Eine Evaluierung am Beispiel objektorientierter Analysemethoden*. Shaker Verlag, Aachen, 1996.

[SW63] C.E. Shannon and W. Weaver. *Mathematical Theory of Communication*. B&T, 1963.

[SW04] P. Soffer and Y. Wand. Goal-driven analysis of process model validity. In A. Persson and J. Stirna, editors, *Advanced Information Systems Engineering, 16th International Conference, CAiSE 2004, Riga, Latvia, June 7-11, 2004, Proceedings*, volume 3084 of *Lecture Notes in Computer Science*, pages 521–535. Springer, 2004.

[Tay11] F.W. Taylor. *The Principles of Scientific Management*. Harper and Brothers, New York and London, 1911.

[TD06a] O. Thomas and T. Dollmann. Fuzzy-EPK-Modelle: Attributierung und Regelintegration. In M. Nüttgens and F.J. Rump and J. Mendling, editor, *Proceedings of the 5th GI Workshop on Business Process Management with Event-Driven Process Chains (EPK 2006)*, pages 49–68, Vienna, Austria, December 2006. German Informatics Society.

[TD06b] W. Trochim and J.P. Donnelly. *The Research Methods Knowledge Base*. Atomic Dog Publishing, 3rd edition, 2006.

[TF06a] O. Thomas and M. Fellmann. Semantic event-driven process chains. In *Semantics for Business Process Management 2006 (SBPM 2006) : Workshop at 3rd European Semantic Web Conference (ESWC 2006)*, Budva, Montenegro, June 2006.

[TF06b] O. Thomas and M. Fellmann. Semantische ereignisgesteuerte prozessketten. In J. Schelp, R. Winter, U. Frank, B. Rieger, and K. Turowski, editors, *Integration, Informationslogistik und Architektur: DW2006*, 2006.

[THA02] O. Thomas, C. Hüsselmann, and O. Adam. Fuzzy-Ereignisgesteuerte Prozessketten - Geschäftsprozessmodellierung unter Berücksichtigung unscharfer Daten. In M. Nüttgens and F.J. Rump, editor, *Proc. of the 1st GI-Workshop on Business Process Management with Event-Driven Process Chains (EPK 2002), Trier, Germany*, pages 7–16, 2002.

[Tho05] O. Thomas. Understanding the term reference model in information systems research: History, literature analysis and explanation. In C. Bussler and

A. Haller, editors, *Business Process Management Workshops, BPM 2005 International Workshops, BPI, BPD, ENEI, BPRM, WSCOBPM, BPS, Nancy, France, September 5, 2005, Revised Selected Papers*, volume 3812 of *Lecture Notes in Computer Science*, pages 484–496, 2005.

[Tja01] G.S. Tjaden. Business process structural analysis. Technical report, Georgia Tech Research Corp., June 2001.

[TM05] F.L. Tiplea and D.C. Marinescu. Structural soundness of workflow nets is decidable. *Inf. Process. Lett.*, 96(2):54–58, 2005.

[TNM96] G.S. Tjaden, S. Narasimhan, and S. Mitra. Structural effectiveness metrics for business processes. In *Proceedings of the INFORMS Conference on Information Systems and Technology*, pages 396–400, May 1996.

[Too04] R. van der Toorn. *Component-Based Software Design with Petri nets: An Approach Based on Inheritance of Behavior*. PhD thesis, Eindhoven University of Technology, Eindhoven, The Netherlands, 2004.

[Tor58] W.S. Torgerson. *Theory and methods of scaling*. Wiley, 1958.

[TS06] O. Thomas and A.-W. Scheer. Tool support for the collaborative design of reference models - a business engineering perspective. In *39th Hawaii International International Conference on Systems Science (HICSS-39 2006), CD-ROM / Abstracts Proceedings, 4-7 January 2006, Kauai, HI, USA*. IEEE Computer Society, 2006.

[Tuk77] J.W. Tukey. *Exploratory Data Analysis*. Addison-Wesley, 1977.

[Ulr49] H. Ulrich. *Betriebswirtschaftliche Organisationslehre. Eine Einführung*. Verlag Paul Haupt, 1949.

[VA00] H.M.W. Verbeek and W.M.P. van der Aalst. Woflan 2.0: A Petri-net-based Workflow Diagnosis Tool. In M. Nielsen and D. Simpson, editors, *Application and Theory of Petri Nets 2000*, volume 1825 of *LNCS*, pages 475–484. Springer-Verlag, Berlin, 2000.

[VA06] H. M. W. Verbeek and W.M.P. van der Aalst. On the verification of EPCs using T-invariants. BPMCenter Report BPM-06-05, BPMcenter.org, 2006.

[Val98] A. Valmari. The state explosion problem. In W. Reisig and G. Rozenberg, editors, *Lectures on Petri Nets I: Basic Models, Advances in Petri Nets, the volumes are based on the Advanced Course on Petri Nets, held in Dagstuhl, September 1996*, volume 1491 of *Lecture Notes in Computer Science*, pages 429–528. Springer, 1998.

[VBA01] H.M.W. Verbeek, T. Basten, and W.M.P. van der Aalst. Diagnosing Workflow Processes using Woflan. *The Computer Journal*, 44(4):246–279, 2001.

[VDMA06] H.M.W. Verbeek, B.F. van Dongen, J. Mendling, and W.M.P. van der Aalst. Interoperability in the ProM Framework. In T. Latour and M. Petit, editors, *Proceedings of the CAiSE Workshops at the 18st Conference on Advanced Information Systems Engineering (CAiSE 2006)*, pages 619–630, Luxembourg, Luxembourg, 2006.

[Ver04] H.M.W. Verbeek. *Verification and Enactment of Workflow Management Systems*. PhD thesis, Eindhoven University of Technology, Eindhoven, The Netherlands, 2004.

[VPAJ07] H.M.W. Verbeek, A.J. Pretorius, W.M.P. van der Aalst, and J.J. van Wijk. Visualizing state spaces with Petri nets. Computer Science Report 07/01, Eindhoven University of Technology, Eindhoven, The Netherlands, 2007.

[VPL07] V. K. Vaishnavi, S. Purao, and J. Liegle. Object-oriented product metrics: A generic framework. *Information Sciences*, 177:587–606, January 2007.

[VW93] P.F. Velleman and L. Wilkinson. Nominal, ordinal, interval, and ratio typologies are misleading. *The American Statistician*, 47(1):65–72, February 1993.

[WAD+05] P. Wohed, W.M.P. van der Aalst, M. Dumas, A.H.M. ter Hofstede, and N. Russell. Pattern-Based Analysis of the Control-Flow Perspective of UML Activity Diagrams. In L. Delcambre, C. Kop, H.C. Mayr, J. Mylopoulos, and

O. Pastor, editors, *24nd International Conference on Conceptual Modeling (ER 2005)*, volume 3716 of *LNCS*, pages 63–78. Springer-Verlag, Berlin, 2005.

[WAD+06] P. Wohed, W.M.P. van der Aalst, M. Dumas, A.H.M. ter Hofstede, and N. Russell. On the Suitability of BPMN for Business Process Modelling. In S. Dustdar, J.L. Fiadeiro, and A. Sheth, editors, *Business Process Management, 4th International Conference, BPM 2006*, volume 4102 of *Lecture Notes in Computer Science*, pages 161–176. Springer-Verlag, September 2006.

[WADH03] P. Wohed, Wil M.P. van der Aalst, M. Dumas, and Arthur H. M. ter Hofstede. Analysis of Web Service Composition Languages: The Case of BPEL4WS. In *Proceedings of Conceptual Modeling - ER 2003*, volume LNCS 2813, pages 200–215, 2003.

[WAHE06] M.T. Wynn, W.M.P. van der Aalst, A.H.M. ter Hofstede, and D. Edmond. Verifying Workflows with Cancellation Regions and OR-joins: An Approach Based on Reset Nets and Reachability Analysis. In S. Dustdar, J.L. Fiadeiro, , and A. Sheth, editors, *Proceedings of BPM 2006*, volume 4102 of *Lecture Notes in Computer Science*, pages 389–394, Vienna, Austria, 2006. Springer-Verlag.

[WEAH05] M.T. Wynn, D. Edmond, W.M.P. van der Aalst, and A.H.M. ter Hofstede. Achieving a General, Formal and Decidable Approach to the OR-join in Workflow using Reset nets. In G. Ciardo and P. Darondeau, editors, *Applications and Theory of Petri Nets 2005*, volume 3536 of *Lecture Notes in Computer Science*, pages 423–443, 2005.

[Wes00] M. Weske. Foundation, Design, and Implementation of Dynamic Adaptations in a Workflow Management System. Fachbericht Angewandte Mathematik und Informatik 6/2000-I, Universität Münster, Münster, Germany, 2000.

[Wey88] E.J. Weyuker. Evaluating software complexity measures. *IEEE Transactions on Software Engineering*, 14(9):1357–1365, 1988.

[WHM06] I. Weber, J. Haller, and J.A. Mülle. Derivation of executable business processes from choreographies in virtual organizations. In *Proc. of XML4BPM 2006*, February 2006.

[Win06] Wintergreen Research. Business process management (bpm) market opportunities, strategies, and forecasts, 2006 to 2012. Technical Report Pub ID: WGR1352720, Wintergreen Research, http://www.marketresearch.com/product/display.asp?productid=1352720&g=1, October 2006.

[Win88] T. Winograd. A language/action perspective on the design of cooperative work. *Human-Computer Interaction*, 3(1):3–30, 1987-88.

[WJH03] B.B. Welch, K. Jones, and J. Hobbs. *Practical Programming in Tcl and Tk*. Prentice Hall International, 4th edition, 2003.

[WKR01] A. Winter, B. Kullbach, and V. Riediger. An Overview of the GXL Graph Exchange Language. In S. Diehl, editor, *Software Visualization - International Seminar Dagstuhl Castle*, volume 2269 of *Lecture Notes in Computer Science*, pages 324–336, 2001.

[WLPS06] U. Wahli, L. Leybovich, E. Prevost, and R. Scher. *Business Process Management: Modeling Through Monitoring*. IBM Redbooks. Vervante, 2006.

[Wor02] Workflow Management Coalition. Workflow Process Definition Interface – XML Process Definition Language. Document Number WFMC-TC-1025, October 25, 2002, Version 1.0, Workflow Management Coalition, 2002.

[Wor05] Workflow Management Coalition. Workflow Process Definition Interface – XML Process Definition Language. Document Number WFMC-TC-1025, October 3, 2005, Version 2.00, Workflow Management Coalition, 2005.

[WVA+06a] M.T. Wynn, H.M.W. Verbeek, W.M.P. van der Aalst, A.H.M. ter Hofstede, and D. Edmond. Reduction rules for reset workflow nets. BPMCenter Report BPM-06-25, BPMcenter.org, 2006.

[WVA+06b] M.T. Wynn, H.M.W. Verbeek, W.M.P. van der Aalst, A.H.M. ter Hofstede, and D. Edmond. Reduction rules for yawl workflow nets with cancellation regions and or-joins. BPMCenter Report BPM-06-24, BPMcenter.org, 2006.

[WW90] Y. Wand and R. Weber. *Studies in Bunge's Treatise on Basic Philosophy*, chapter Mario Bunge's Ontology as a Formal Foundation for Information Systems Concepts, pages 123–149. the Poznan Studies in the Philosophy of the Sciences and the Humanities. Rodopi, 1990.

[WW95] Y. Wand and R. Weber. On the deep structure of information systems. *Information Systems Journal*, 5:203–223, 1995.

[WW02] Y. Wand and R. Weber. Research Commentary: Information Systems and Conceptual Modeling - A Research Agenda. *Information Systems Research*, 13(4):363–376, 2002.

[YC79] E. Yourdon and L.L. Constantine. *Structured Design*. Prentice Hall, 1979.

[ZHB+06] W. Zhao, R. Hauser, K. Bhattacharya, B.R. Bryant, and F. Cao. Compiling business processes: untangling unstructured loops in irreducible flow graphs. *Int. Journal of Web and Grid Services*, 2(1):68–91, 2006.

[Zis77] M.D. Zisman. *Representation, Specification and Automation of Office Procedures*. PhD thesis, University of Pennsylvania, Warton School of Business, 1977.

[Zis78] M. D. Zisman. Use of production systems for modeling asynchronous concurrent processes. *Pattern-Directed Inference Systems*, pages 53–68, 1978.

[ZM05] J. Ziemann and J. Mendling. EPC-Based Modelling of BPEL Processes: a Pragmatic Transformation Approach. In *International Conference "Modern Information Technology in the Innovation Processes of the Industrial Enterprises*, Genova, Italy, 2005.

[ZR96] O. Zukunft and F.J. Rump. *Business Process Modelling*, chapter From Business Process Modelling to Workflow Management: An Integrated Approach, pages 3–22. Springer-Verlag, 1996.

[Zus91] H. Zuse. *Software Complexity: Measures and Methods*. Walter de Gruyter and Co, New Jersey, 1991.