

# Requirements Lifecycle Management and Release Planning in Market-Driven Requirements Engineering Processes

Pär Carlshamre<sup>1</sup> and Björn Regnell<sup>2</sup>

<sup>1</sup>Research and Innovation, Ericsson Radio Systems, Linköping, Sweden, par.carlshamre@ericsson.com

<sup>2</sup>Dept. of Communication Systems, Lund University, Sweden, bjorn.regnell@telecom.lth.se

## Abstract

*In market-driven software evolution, the objectives of a requirements engineering process include the continuous management of new and changed requirements in a way that ensures competitiveness on the market place. This paper compares two independently developed industrial market-driven requirements engineering processes, which both apply continuous requirements management using state-oriented life cycle models in the fostering of requirements from invention to release. Similarities and differences between the models are identified, and opportunities of lifecycle-oriented requirements management are recognized. The challenge of release planning is elaborated, with a particular focus on the crucial task of managing requirements dependencies.*

## 1. Introduction

When developing software for a market-place, rather than for a single customer based on a contract, the pressure on short time-to-market is evident [10, 7]. The achievement of high software quality in a market-driven context is inherently related to the challenge of meeting customers' demands [4]. Market-driven Requirements Engineering processes [1, 3] have a strong focus on requirements prioritization [6, 5] and often deliver incremental releases of a continuously evolving product [8].

This paper describes and compares two RE processes used at different companies (Ericsson Radio Systems AB and Telelogic AB), which both address the challenges of market-driven software development. Traditional phase-oriented process models are replaced by a process model with a concurrent approach, where individual requirements are fostered in parallel from invention to release. The fostering of requirements is managed based on a lifecycle model, which defines a ladder of states for each requirement to climb.

The two companies have developed their own market-driven RE processes independently. Although their respective product domains and market situations are different, the processes are quite similar.

Telelogic AB is a fast-growing medium-sized CASE-tool vendor. Telelogic's main product family is called Telelogic Tau; a software development environment for real-time systems, used by many of the world's largest telecommunica-

tion systems providers in their software development. In 1995, Telelogic had problems with delayed product releases and an improvement programme was launched with focus on requirements management. Within this programme, REPEAT (Requirements Engineering Process At Telelogic) was developed, and after its introduction a significant improvement in delivery precision and product quality was gained [9].

Ericsson Radio Systems AB develops software for operation and maintenance of mobile telephone networks. The customers are mobile network operators of various sizes on almost all continents. A major part of operators' costs relate to operation, maintenance and administration, and software solutions for supporting these activities demand continuous improvement. An improvement programme was launched in early 1998 with the aim of reducing lead time by an order of magnitude while increasing productivity and quality. One of the activities within this framework was to develop RDEM (Requirement Driven Evolution Model).

In Sections 2 and 3, the RDEM and REPEAT processes are described respectively. The processes are compared in Section 4, and Section 5 outlines major opportunities and challenges.

## 2. The RDEM process

A requirement in RDEM is defined as a hierarchical structure of attributes. An attribute can be either simple or complex (i.e., containing attributes). It can also be either optional or mandatory, as decided by the product management. Such decisions are made as close to the actual development as possible, and a specific attribute structure is defined for each product. Software development is viewed as a learning process, where it is known in advance what should be learned about the requirements. Thus, the development is conceptualized as assigning values to attributes, and the actual procedures can be chosen in accordance with local expertise, tradition and preferences.

The overall process is divided into three sub-processes, that are continuous and asynchronous: Capturing and Specification results in specified requirements residing in the database. The Planning sub-process selects specified requirements, adds production information, and creates bundles of requirements, which together with an old increment

is realized into a new increment by the Realization sub-process.

## 2.1 Roles and responsibilities

The main organizational unit for a development effort is called an Application Domain Team. This represents a step away from Ericsson’s traditional line/project matrix organization, and the intent is to tie people closer to a product in order to keep and nourish product knowledge.

RDEM specifies more than 20 roles in terms of e.g., responsibilities and authorities, required competence and communication paths. One person can have more than one role, and only a few roles are mandatory for an Application Domain Team: Domain Manager, Architect, Development Manager, Lead Analyst, Lead Designer and Test Leader. The first four roles, together with a number of stakeholders (e.g. customer representatives, executive management) forms the Product Committee (PC), which is responsible for all major decisions. In particular, they are responsible for the decision whether a requirement should be propagated to the next state in the requirements lifecycle model.

## 2.2 The RDEM Lifecycle Model

Progression of requirements in RDEM is defined in terms of what is known about the requirements. This knowledge is contained in the requirements attributes, and the state model can be viewed as a definition of certain levels of knowledge. The state model is designed to reflect the natural stages in a development process, and the decision points between states can be viewed as milestones. There are four defined states:

**Captured.** In RDEM, nothing is a requirement until it is captured. In effect, the Product Committee defines what are requirements and what are not. The decision that a requirement is captured represents a commitment and an intention to further elaborate the requirement.

**Specified.** Specified requirement holds all information needed to proceed with implementation and verification from a narrow, system-oriented perspective. However, when

a requirement becomes specified, it does not yet hold any production-oriented information, e.g., when and how the requirement is best implemented from a customer or product management perspective. At this stage requirements are grouped into requirements bundles called *deltas*. A delta holds a specific set of attributes, i.e., information tied to the collection of requirements, rather than to the individual requirements. These attributes should be assigned values, before the delta and its constituting requirements can be considered to be Planned.

**Planned.** All items that are elevated to this state are deltas. A delta can, however, in special cases contain only one requirement. Only planned deltas can be implemented and verified (realized, in RDEM terminology). Realization represents the execution phase of a small project, and to give an idea about its size, it should require less or equal to the effort of 10 people working for two months. As the realization processes are carried out, further information that is useful for the ongoing realization work can be added to the delta, such as the actual code, test reports and configuration information. When a delta has been implemented and verified, and the Product Committee has agreed that is the case, the delta is moved to the Realized state.

**Realized.** When a delta becomes realized together with the existing design base, it is termed an increment. From an RDEM perspective, this is the final state.

At each decision point, the Product Committee can select between the following two alternatives:

- *Propagate the item to the next state.* This implies that the Product Committee confirms that all necessary information is available, and that the requirement (or delta) should be further elaborated (or delivered).
- *Not propagate the item.* This means that some issues need to be further investigated before a decision can be made.

The state model is not rigid in terms of what information to be added in what state. The states only define a least acceptable level of knowledge about the requirements.

## 3. The REPEAT process

REPEAT manages requirements continuously by controlling a product pipe-line in which three releases are developed in parallel. The product pipe-line delivers a new product release every six months. REPEAT covers typical RE activities, such as elicitation, documentation, and validation, and has a strong focus on requirements selection and release planning.

### 3.1 Roles and responsibilities

There are a number of actors involved in REPEAT. The *Requirements Management Group* (RQMG) is responsible for requirements management, and makes decisions on which requirements to implement. It is also responsible for requirements change management. RQMG includes, among

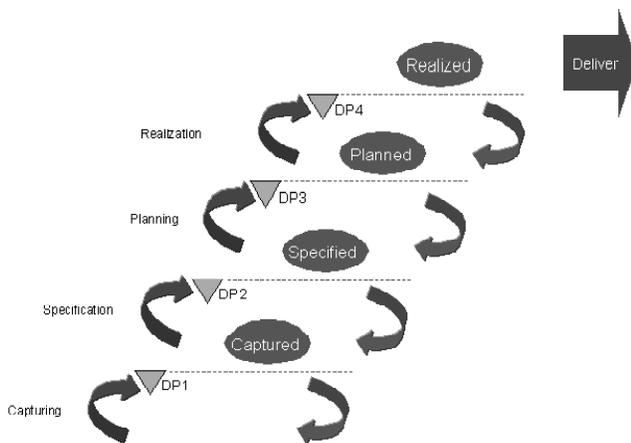


Figure 1. The RDEM requirements lifecycle model.

others, product and project managers together with the quality manager. *Customers & users* provide input and feedback to an *issuer* regarding user and market needs. An issuer can be any employee at Telelogic who submits a requirement to RQMG. An issuer is usually a person from marketing & sales or customer support, but can also be e.g. a developer or a tester. *Requirements Teams* have the responsibility of analysing and specifying a set of requirements. RQMG has several of these teams at their disposal. A requirements team is cross-functional and includes persons participating in implementation, testing, marketing & sales, and customer support. Specific *Experts* are assigned to evaluate a specific requirement in depth, concerned with e.g. cost estimation and impact analysis. *Construction Teams* have the responsibility of designing and implementing a set of requirements. *Test Teams* have the responsibility of verifying a set of requirements. The *Requirements Database* (RQDB) is an in-house-built database system for storing all requirements. RQDB has a web-interface that can be accessed by Telelogic employees from a multi-continent intranet.

### 3.2 The REPEAT Lifecycle Model

The semantics of the states in the REPEAT lifecycle model are explained below. The RQMG with support from experts is responsible for deciding on requirement state transitions.

**New.** This state represents the initial state of a requirement, and every requirement is defined as new immediately after it has been issued and given an initial priority.

**Assigned.** A requirement is elevated to the assigned state when an expert has been assigned to investigate the requirement and determine the value of a number of attributes.

**Classified.** When reaching this state, an expert has assigned values to attributes representing a rough estimate of cost and architectural impact. Comments and implementation ideas may also be stated.

**Selected.** All requirements in this state are selected for implementation for the coming release. They are sorted in priority order on two list: a must-list for mandatory requirements and a wish-list for “nice-to-have” requirements. They also have attributes assigned concerning detailed cost and impact estimations. There is also a more detailed textual

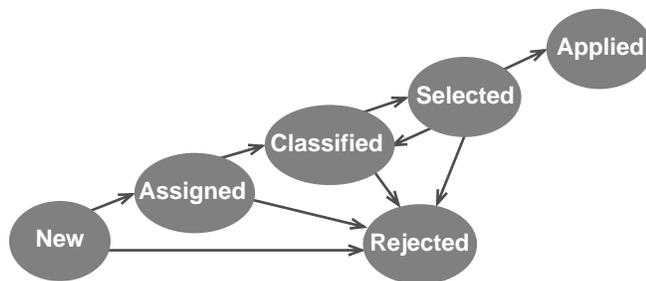


Figure 2. The REPEAT requirements lifecycle model.

specification of the requirement. A selected requirement may be de-selected, due to changed circumstances, and then re-enters the classification state or gets rejected.

**Applied.** This is an end-state indicating that the requirement has been implemented and verified. The requirement is now incorporated in a release that can be marketed to customers.

**Rejected.** This is an end-state indicating that the requirement has been rejected, e.g. because it is a duplicate, already implemented, or it does not comply with the long-term product strategy.

REPEAT is instantiated for each release, and each process instance has a fixed duration of 14 months. A new product version is released at fixed dates every six months, which implies that different process instances overlap with three simultaneous enactments. Each process instance passes specific milestones at pre-defined dates. A more detailed description of REPEAT can be found in [9].

## 4. Similarities and differences

Considering that the two models have been developed in parallel but completely independently, they are quite similar in many respects. Two areas with salient similarities are elaborated below.

**Lifecycle models.** Both RDEM and REPEAT have lifecycle models with states representing the refinement level of each individual requirement in its progress towards release. The states defined by the two models are quite similar. The *captured* state in RDEM, roughly corresponds to the *assigned* state of REPEAT. The *realized* and *applied* states have identical semantics, representing a successful implementation of the requirement. Both the *planned* and *selected* states represent that the requirement is approved to be implemented. The states *specified* and *classified* is in both models reached when certain detailed information is available for the requirement.

**Roles and responsibilities.** Both RDEM and REPEAT have a strong focus on roles and responsibilities, and in both cases they are explicitly specified in detail. When the phase-oriented development model is replaced by an asynchronous fostering of requirements through a life-cycle, it becomes even more important that responsibilities are well defined, which is recognized by both organizations.

Beyond minor discrepancies on a detailed level, there is a general difference in the driving forces behind the two models. It may be stated that Telelogic’s REPEAT is more focused on time-to-market, whereas Ericsson’s RDEM is more focused on quality. Some salient differences, aligned with the different driving forces, are described below.

**Types of attributes.** There are fewer defined requirements attributes in the REPEAT case, based on a major objective of “keeping it simple” to allow quick action. The REPEAT attributes are mainly production-oriented (e.g.,

priority, needed effort, must/wish categories), while the RDEM attributes have a stronger focus on quality issues, such as usability (e.g., user categories, user goal description, impact on operator procedures).

**Release scheduling.** RDEM does not prescribe a fixed release schedule as in the REPEAT case. In the RDEM approach, release dates are set based on what is planned to be delivered. In REPEAT, on the other hand, what can be delivered is governed by fixed release dates.

**Requirements bundling.** The issue of how to bundle requirements into releases are treated differently by the two processes. In RDEM requirements bundles are made explicit by the concept of deltas, which are propagated through the lifecycle as first-class objects. In REPEAT, however, the bundling of requirements are made in the process of selecting requirements for the coming release by the use of so called must- and wish-lists. The sum of the estimated effort for the implementation of the must-list shall not exceed 70%, and the sum of the estimated effort for the implementation of the wish-list shall not exceed 60% of the available effort. This implies a risk buffer of 30% for the must-list. (More information on the must- and wish-lists is available in [9]).

Considering the two companies' different market situations, the differences are understandable. In a true market-driven situation time-to-market is considered to be more important than quality [11], and REPEAT can be considered to take the market driven approach further than RDEM, simply because Telelogic has a larger number of customers, the customers are more anonymous, there is only one major product, etc. The customers of Ericsson Radio, on the other hand (mobile network operators), are well known, the application domain is more specialized, the number of competing products are fewer, etc. Telelogic's market is hence closer to pure "off-the-shelf" compared to Ericsson Radio Systems'.

## 5. Opportunities and challenges

Beyond the fact that the lifecycle approach described here has indicated improvements in release precision and product quality [13], there are further potential benefits. Some of these opportunities are elaborated below.

**Methodology independence.** The lifecycle view represents a methodology-independent approach to RE, in that it focuses on results rather than on activities. It is reasonable to believe that this makes introduction and deployment easier in a company with hundreds of development teams (with their own method flavours) spread all over the world. In the Ericsson case, a major reason for introducing a lifecycle approach is the fact that software developers are tired of having new methods thrown at them every second year or so. They usually have local methods, based on the best experiences from many different methodologies.

**Balanced quality.** An attribute-centric view may help tackle the problems related to the plentitude of specific methods and techniques (sub-)optimizing specific quality aspects; there are specific usability oriented methods, specific techniques to ensure re-usability, portability, fault tolerance, delivery precision etc. All these methods rest on different assumptions, prioritizations, perspectives and even philosophies. What method is applied in a given case is sometimes more dependent on, e.g., current trends or the loudest voice approach [11], than rational and careful weighing of pros and cons. By deciding on an attribute structure incorporating attributes for all important quality attributes, instead of specifying a methodology, it may be easier to integrate activities with different foci from different methodologies.

**Progress indication.** The states in the lifecycle model indicate progress, in terms of how much is known about the requirements. Progress can also be measured in between states. Given a fixed attribute structure, the ratio between what is known and what is yet to be investigated is evident at all times, and can be used as an indication of progress. As an example, consider the case where a given attribute structure contains 28 leaf-level attributes of which 7 have been set. This approximate to a 25% fulfilment. An interesting area for further research is to apply more elaborated cost models in this context.

**Flexibility.** The lifecycle approach and the attribute-centric view allows the introduction of an adaptable infrastructure for requirements management, where attributes may be introduced when needed, and requirements can change state in the lifecycle model based on changes in the market situation.

There are a number of challenges for organizations who would like to introduce a lifecycle approach similar to what is described here. One of the most important, and complex, tasks in market-driven RE is *release planning* [11, 2]. It is important because it deals with the essential questions of what should be developed when. It is complex because requirements usually have many dependencies. Therefore it is not possible, for example, to just select a subset of the requirements in the database having the highest priority. Important challenges in release planning are elaborated below.

**Requirements dependencies.** When requirements are fostered asynchronously in a lifecycle model, the management of requirements dependencies is central as it deals with how to connect the fragments of information. An important issue is to investigate the different types of dependencies. One way of classifying dependencies using the dimensions of *scope* and *explicitness* is presented in Table 1.

The scope dimension can be divided into internal and external dependencies. Internal dependencies are functions from requirements to requirements. External dependencies

**Table 1.** Two dimensions of requirements dependencies

		Explicitness	
		Explicit	Implicit
Scope	Internal	Preconceived relations between requirements explicitly captured by link attributes. <i>Example:</i> R1 is a refinement of R2.	Coherent subsets of requirements defined by relations between attribute values. <i>Example:</i> R1 and R2 both have customer value > 70%.
	External	Preconceived relations from requirements to entities outside the set of requirements explicitly captured by link attributes. <i>Example:</i> R1 is issued by stakeholder X.	Coherent subsets of requirements defined by relations between properties of entities external to the set of requirements. <i>Example:</i> Both R1 and R2 are strategic to market segments which currently have high sales and revenue

are functions from requirements to entities outside the set of requirements (e.g. stakeholders, delivery dates).

The explicitness dimension can be divided into explicit and implicit dependencies. Explicit dependencies imply that there is a preconceived representation of the dependency, i.e. as an explicit link in the database. Implicit dependencies are such that they only appear dynamically (e.g., through a database query).

Additional dependency types may be identified, which also can support release planning, including *temporal* (“X must be done before Y”), *logical* (“X will not work without Y”), and *value-related* (“the cost of X increases if Y is implemented”). Planning an increment inevitably involves dealing with all these categories. This gives an indication of the complexity of release planning, and further research in the area of requirements dependencies is important.

**Synthesizing priorities.** Another challenge is related to requirements prioritization. When a market place is world wide, it is vital to collect needs and opportunities from different market segments spread around the world. An important challenge is how to synthesize this information before planning the requirement strategy for the next release. Recent studies indicate that distributed prioritization involving representatives for multiple market segments give valuable decision support, and there are many opportunities of further research in this area [12].

**Definition of attributes and states.** In general it is difficult to tell in advance which requirements attributes and lifecycle states fit best in what situation. Experiences from applications of attribute-driven and lifecycle-oriented RE may give valuable insight into this challenge. Telelogic is currently revising the REPEAT lifecycle model and attribute structure, and additional states are under consideration.

Some of the states may be split into more detailed states, and a number of attributes may be added according to the feedback from different stakeholders in the process. In the RDEM case, attributes and states are also adjusted based on lessons learnt from a number of pilot studies.

### Acknowledgements.

The authors would like to give a special acknowledgement to all persons involved in the development of RDEM and REPEAT, especially Kristian Sandahl, Ulrik Pettersson at Ericsson Radio Systems, and Per Beremark, Ola Eklundh, and Anders Ek at Telelogic.

### 6. References

- [1] Carmel, E., Becker, S., “A Process Model for Packaged Software Development”, *IEEE Transactions on Engineering Management*, Vol. 42 No. 1, pp. 50-61, February 1995.
- [2] Deifel, B., “A Model for Version Planning of CCOTS”, *Proceedings of Workshop on Software Change and Evolution (SCE’99)*, Co-Located with ICSE’99, Los Angeles, USA, IEEE Computer Society Press, May 1999.
- [3] Deifel, B., “A Process Model for Requirements Engineering of CCOTS”, *Proceedings of Workshop on Requirements Engineering Process (REP’99)*, Co-located with DEXA’99, Florence Italy, IEEE Computer Society Press, September 1999.
- [4] Hutchings, A., Knox, S., “Creating Products Customers Demand”, *Communications of the ACM*, Vol. 38 No. 5, pp. 72-80, May 1995.
- [5] Karlsson, J., Ryan, K., “A Cost-Value Approach for Prioritizing Requirements”, *IEEE Software*, September/October 1997.
- [6] Lubars, M., Potts, C., Richter, C., “A Review of the State of the Practice in Requirements Modeling”, *Proceedings of First IEEE International Symposium on Requirements Engineering (RE’93)*, San Diego USA, IEEE Computer Society Press, January 1993.
- [7] Novorita, R., Grube, G., “Benefits of Structured Requirements Methods for Market-Based Enterprises”, *Proceedings of International Council on Systems Engineering Sixth Annual International Symposium on Systems Engineering: Practice and Tools (INCOSE’96)*, Boston USA, July 1996.
- [8] Potts, C., “Invented Requirements and Imagined Customers: Requirements Engineering for Off-the-Shelf Software”, *Proceedings of Second IEEE International Symposium on Requirements Engineering (RE’95)*, York UK, IEEE Computer Society Press, March 1995.
- [9] Regnell, B., Beremark, P., Eklundh, O. “A Market-Driven Requirements Engineering Process – Results from an Industrial Process Improvement Programme”, *Journal of Requirements Engineering*, Vol. 3 No. 2, pp. 121-129, 1998.
- [10] Sawyer, P., Sommerville, I., Kotonya, G., “Improving Market-Driven RE Processes”, *Proceedings of International Conference on Product Focused Software Process Improvement (PROFES’99)*, Oulu Finland, June 1999.
- [11] Yeh, A., “Requirements Engineering Support Technique (REQUEST) – A Market Driven Requirements Management Process”, *Proceedings of Second Symposium of Quality Software Development Tools*, pp. 211-223, New Orleans USA, IEEE Computer Society Press, May 1992.
- [12] Regnell, B., Höst, M., Natt och Dag, J., Beremark, P., Hjelm, T. “Visualization of Agreement and Satisfaction in Distributed Prioritization of Market Requirements”, Submitted to REFSQ’2000.