technische universiteit eindhoven

Inheritance of Dynamic Behavior in UML

Wil van der Aalst

Eindhoven University of Technology
Department of Information and Technology
P.O. Box 513, 5600 MB Eindhoven
The Netherlands
w.m.p.v.d.aalst@tm.tue.nl

Outline

1. Motivation
2. Inheritance of behavior
3. Inheritance preserving transformation rules
4. Inheritance of behavior in UML
   - Sequence diagrams
   - Statecharts diagrams
   - Activity diagrams
5. Conclusion

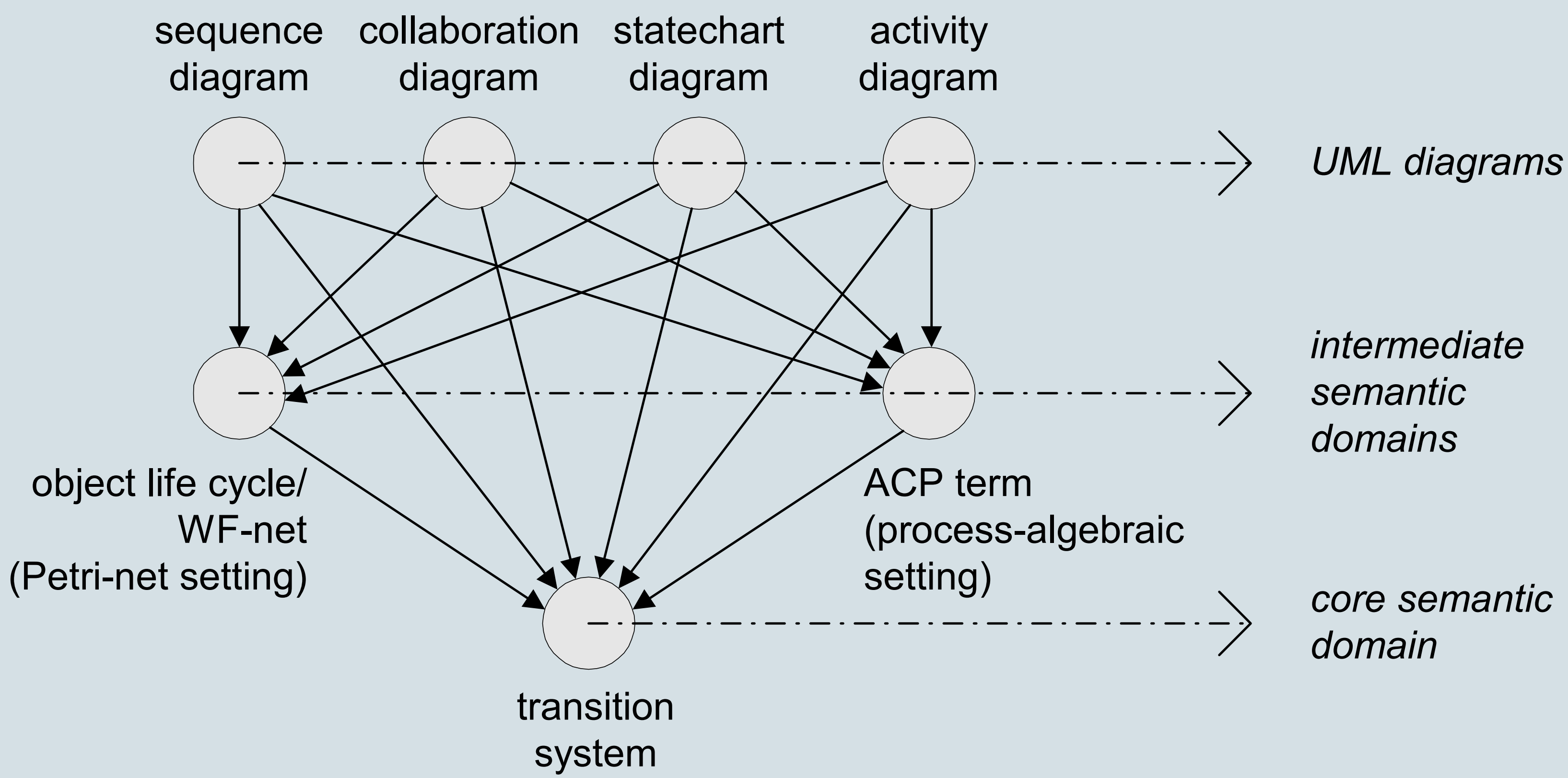
technische universiteit eindhoven

Motivation

- UML has become the standard object-oriented framework.
- Inheritance is one of the cornerstones of object orientation
- UML has at least four diagrams focusing on dynamic behavior / process modeling.
- Yet inheritance is typically restricted to static aspects.
- Frustration: Our work (with Twan Basten and Eric Verbeek) on inheritance has not been adopted by people working on UML.

technische universiteit eindhoven

Approach [Engels et al. 2001]



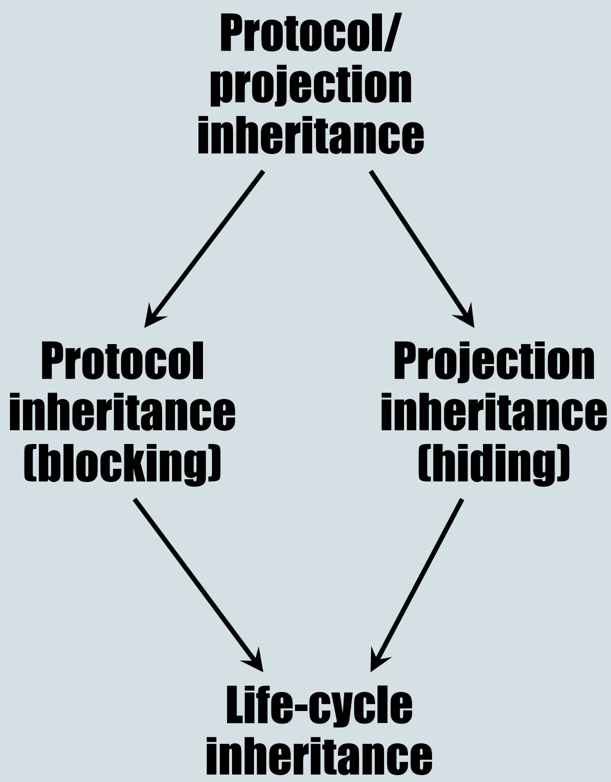
- It is not our aim to provide formal semantics for UML.
- The mappings may be partial/abstractions.
- The intermediate domains are used for analysis purposes.

Inheritance of dynamic behavior

- When is a object life-cycle a subclass of another object life-cyle?
- Four notions of inheritance based on two orthogonal mechanisms.
- **Blocking**: *If it is not possible to distinguish the behaviors of x and y when only methods of x that are also present in y are executed, then x is a subclass of y.* (encapsulation)
- **Hiding:***If it is not possible to distinguish the behaviors of x and y when arbitrary methods of x are executed but when only the effects of methods that are also present in y are considered, then x is a subclass of y.* (abstraction)

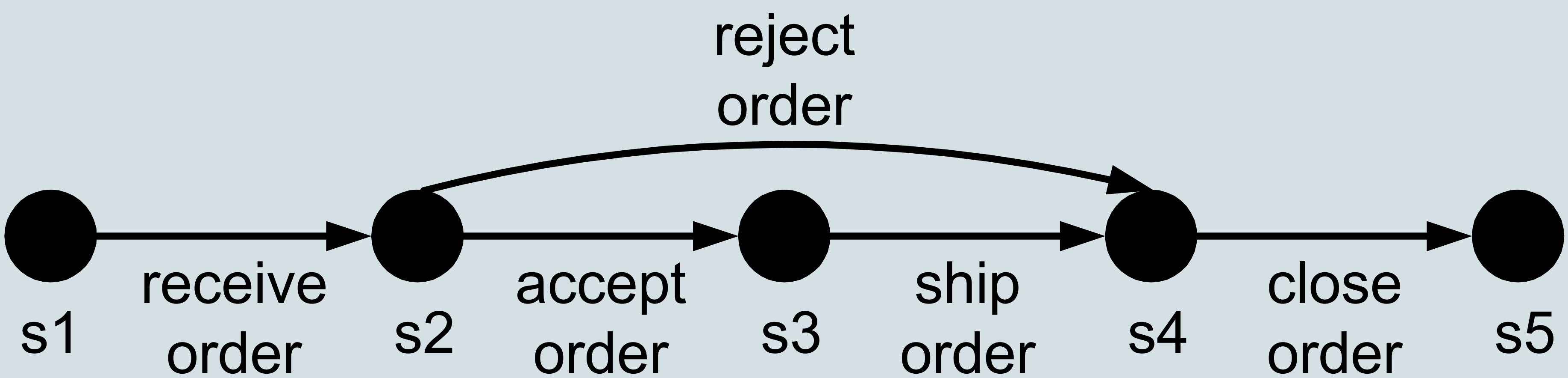
technische universiteit eindhoven

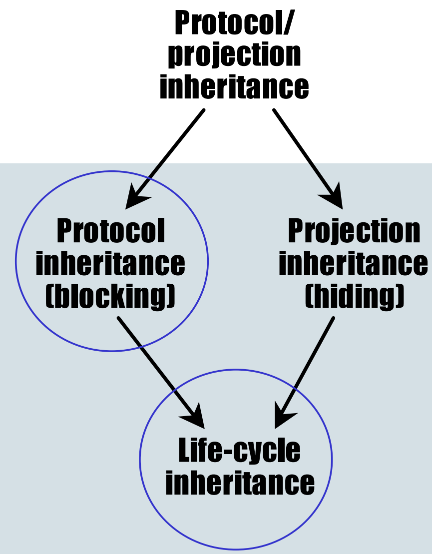
Four notions of inheritance



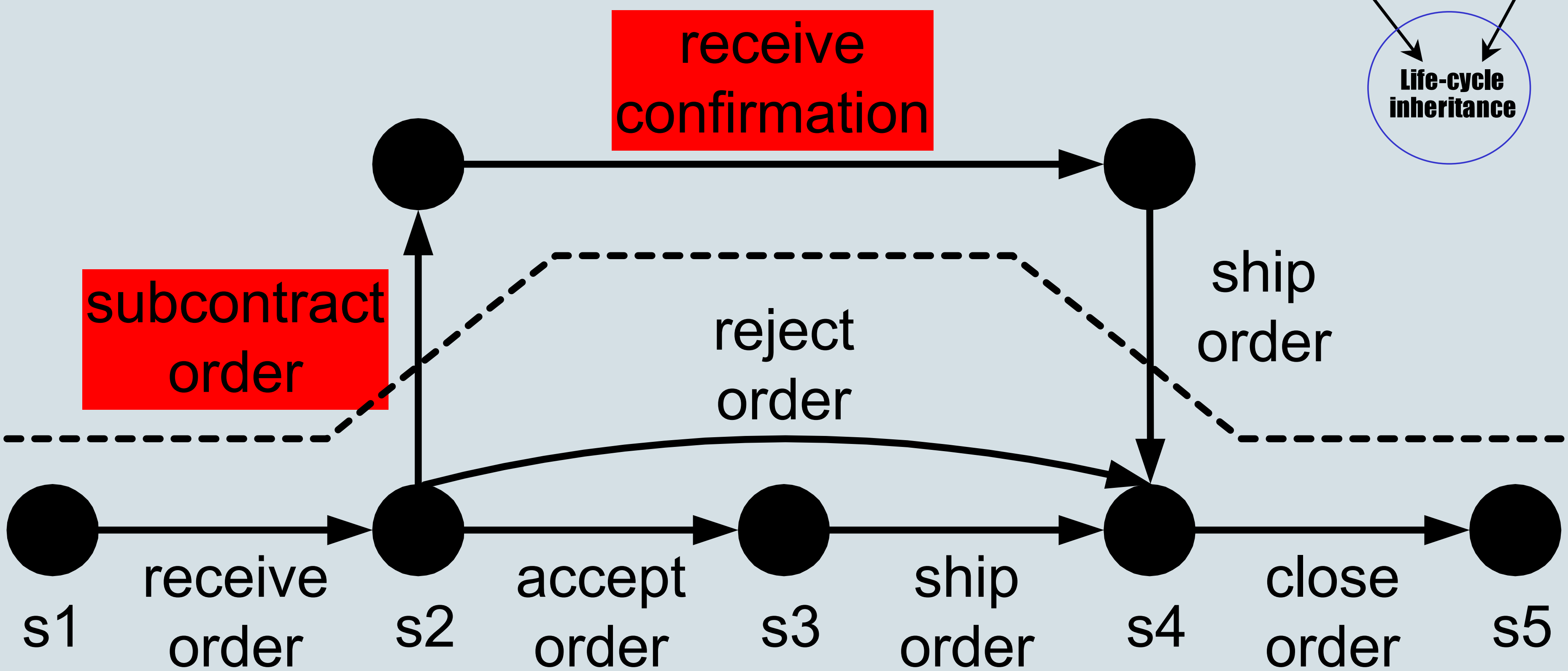
- Have been defined for the core semantic domain (labeled transition systems) and two intermediate semantic domains (Petri nets and ACP).
- We will illustrate the four notions of inheritance using the core semantic domain

Superclass TS1



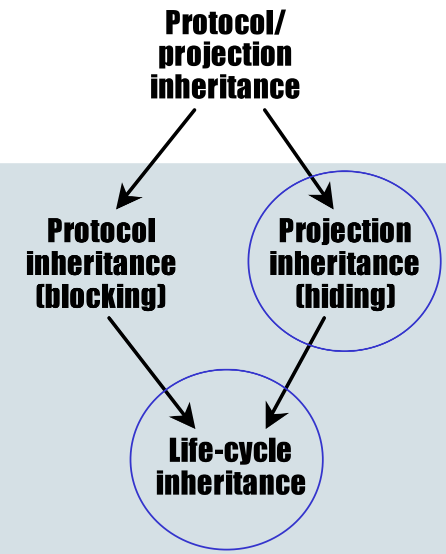


Subclass TS2

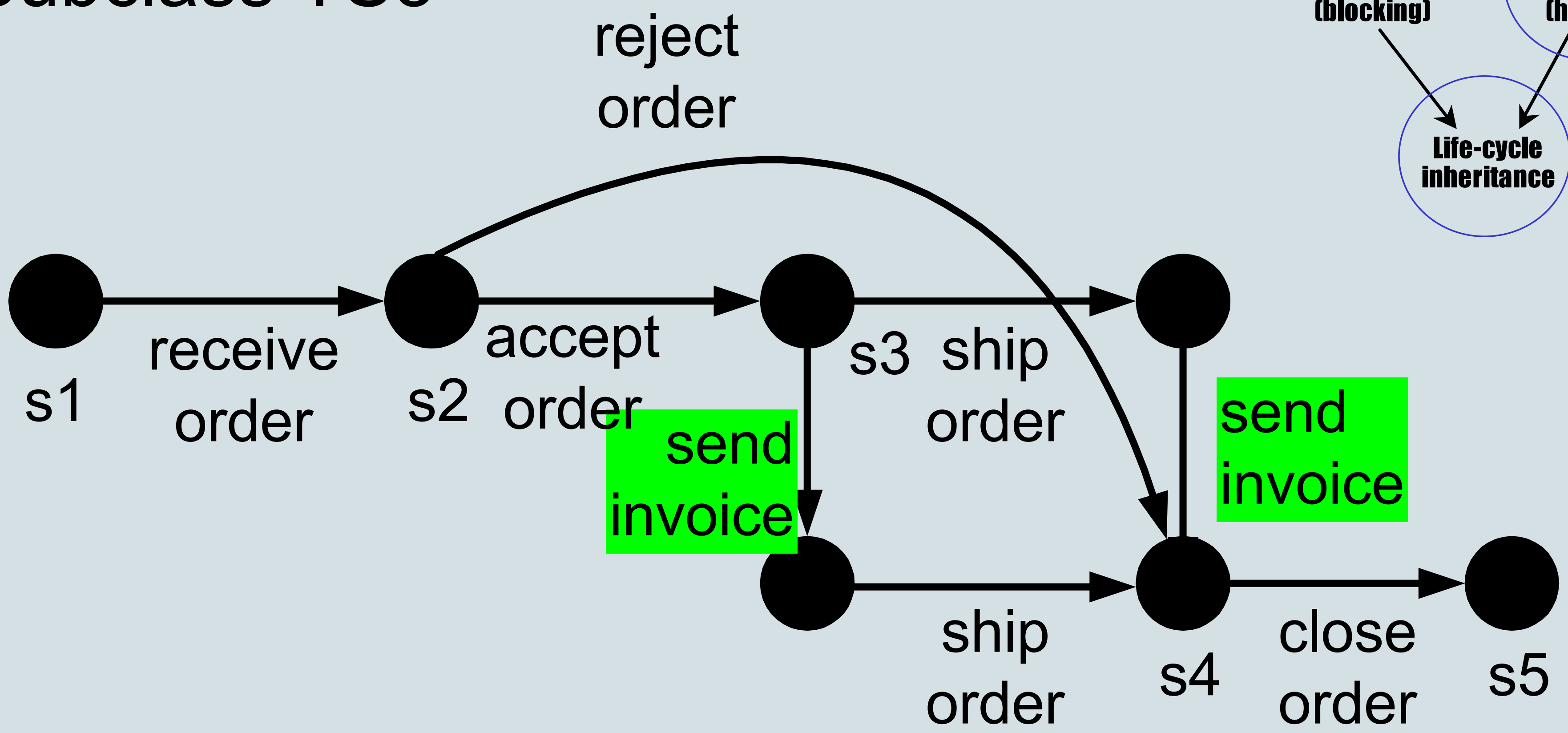


Blocking: *If it is not possible to distinguish the behaviors of x and y when only methods of x that are also present in y are executed, then x is a subclass of y.*

technische universiteit eindhoven



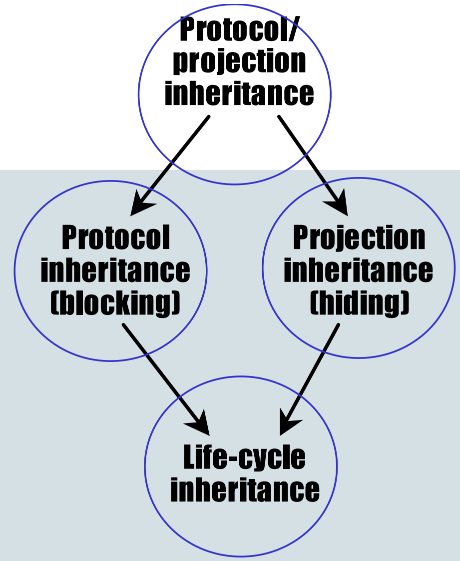
Subclass TS3

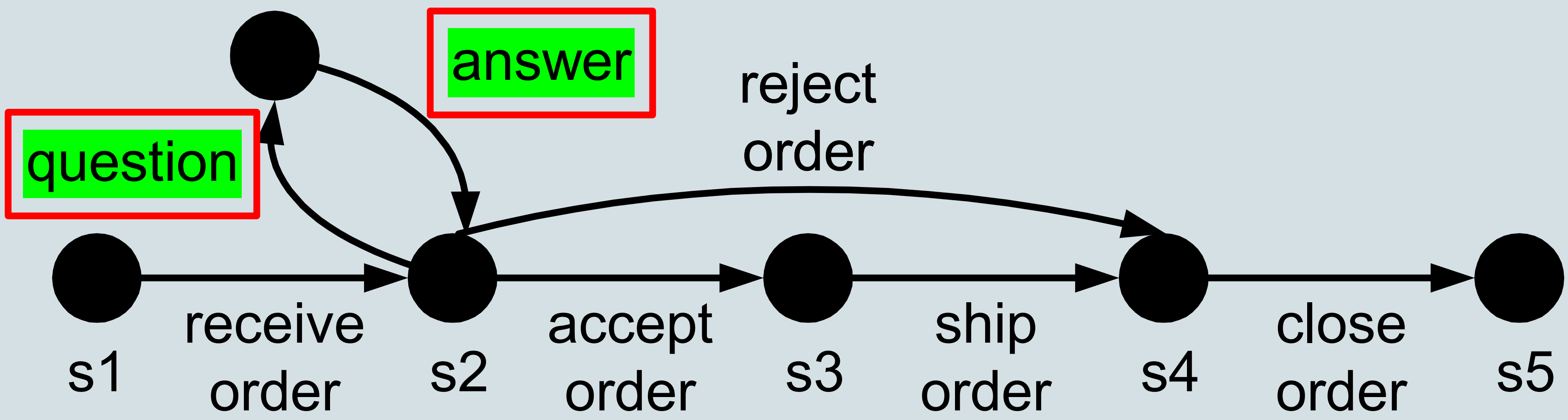


Hiding: *If it is not possible to distinguish the behaviors of x and y when arbitrary methods of x are executed but when only the effects of methods that are also present in y are considered, then x is a subclass of y.*

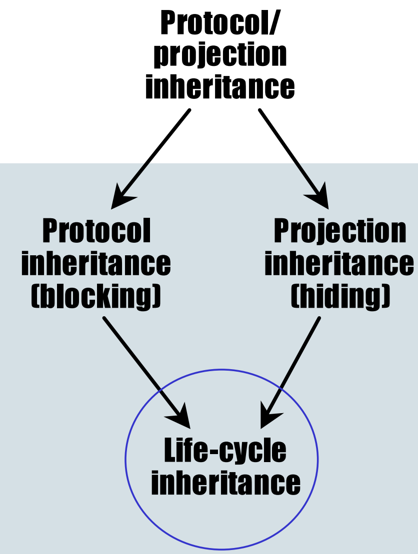
technische universiteit eindhoven

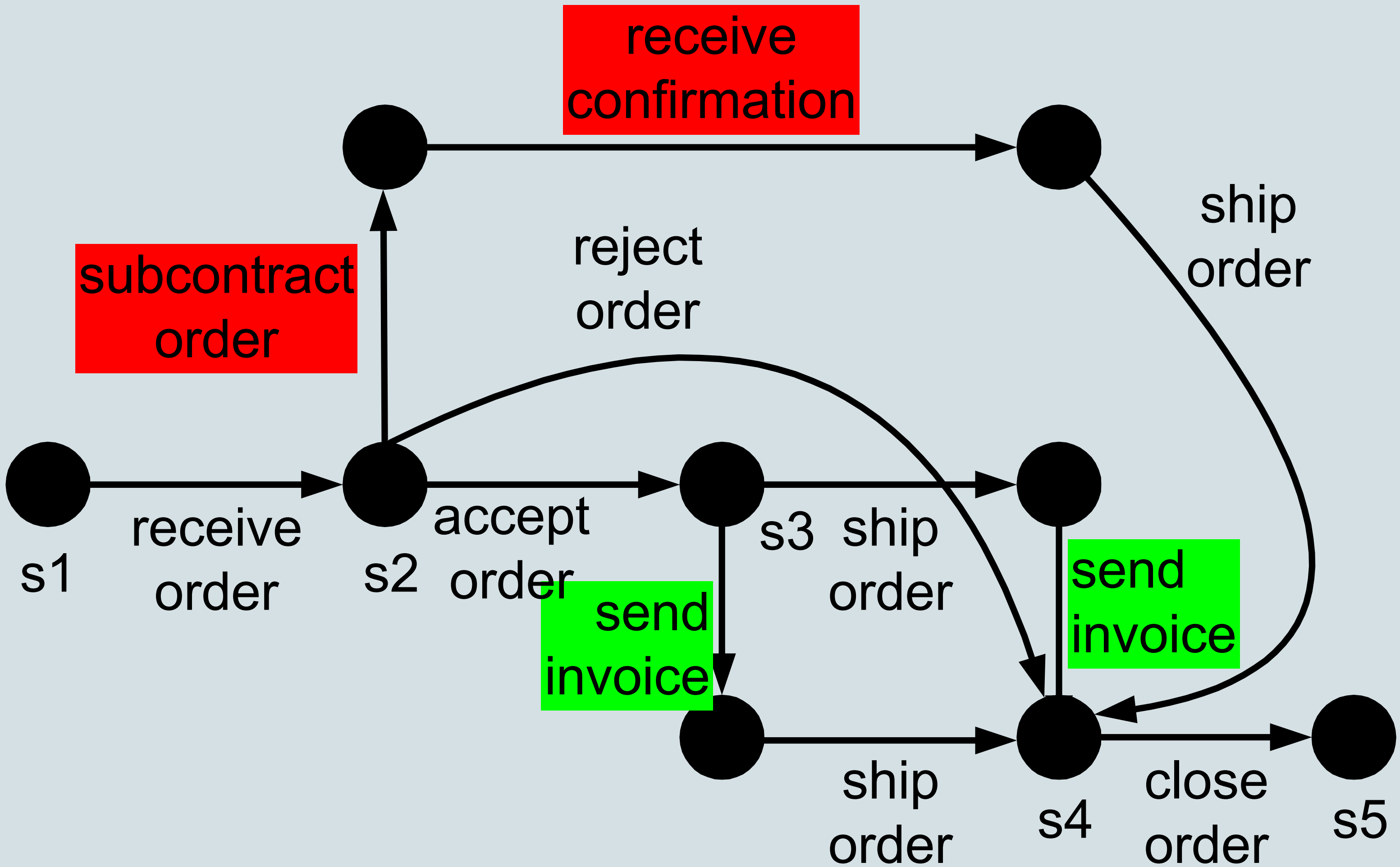
Subclass TS4





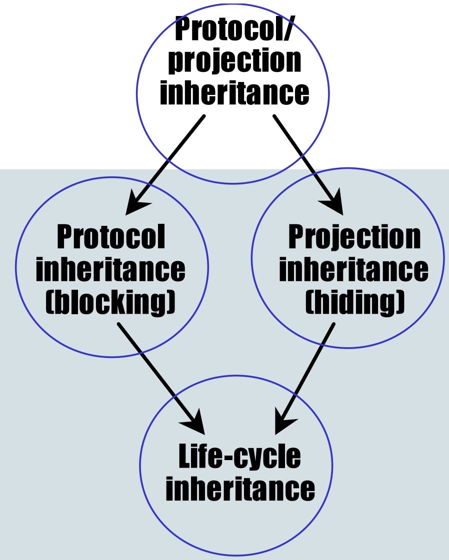
Subclass TS5



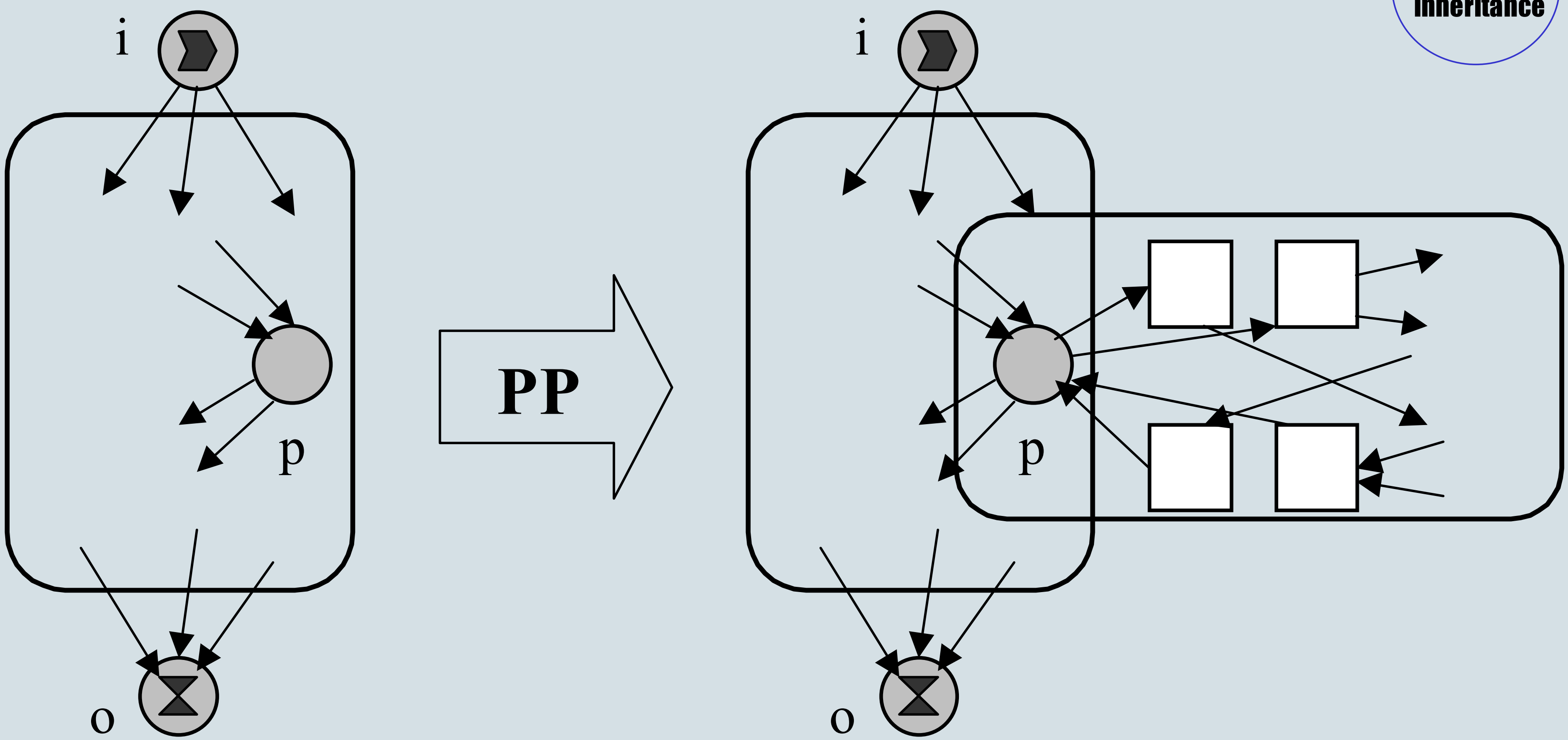


Inheritance preserving transformation rules

- Constructions which preserve one or more notions of inheritance, i.e., rules to transform a superclass into a subclass.
- The four basic rules PP, PT, PJ and PJ3 have been defined in both a Petri-net and a process-algebraic setting (i.e., both intermediate semantic domains considered).
- In this talk we show the rules in a Petri-net setting.
- The requirements for the rules can be checked locally.
- The transformation rules have been equipped with transfer rules to migrate objects from a superclass to a subclass and vice versa.

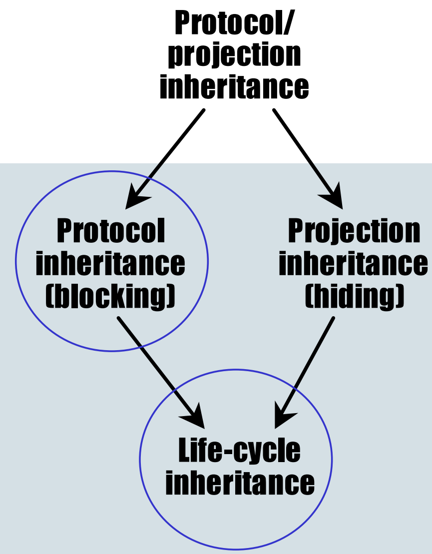


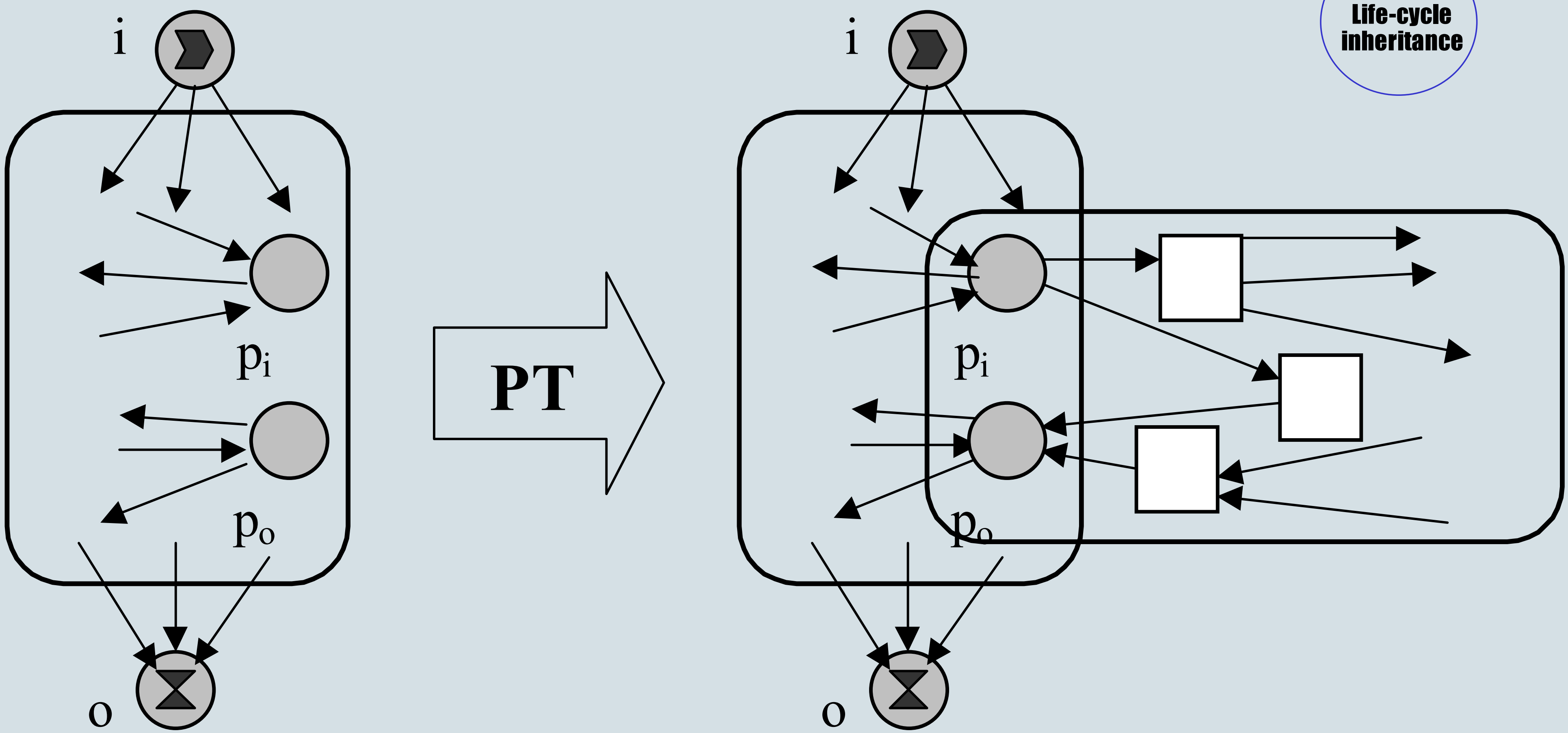
Transformation rule PP: adding loops



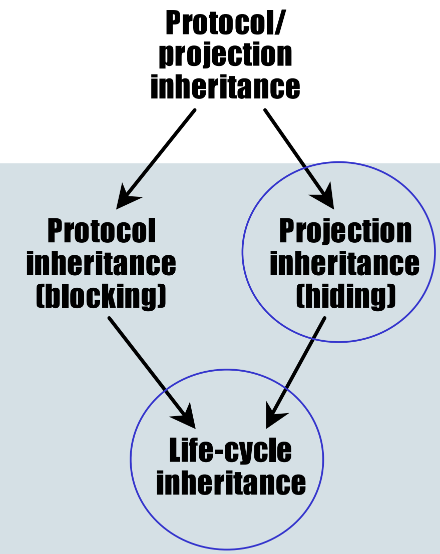
technische universiteit eindhoven

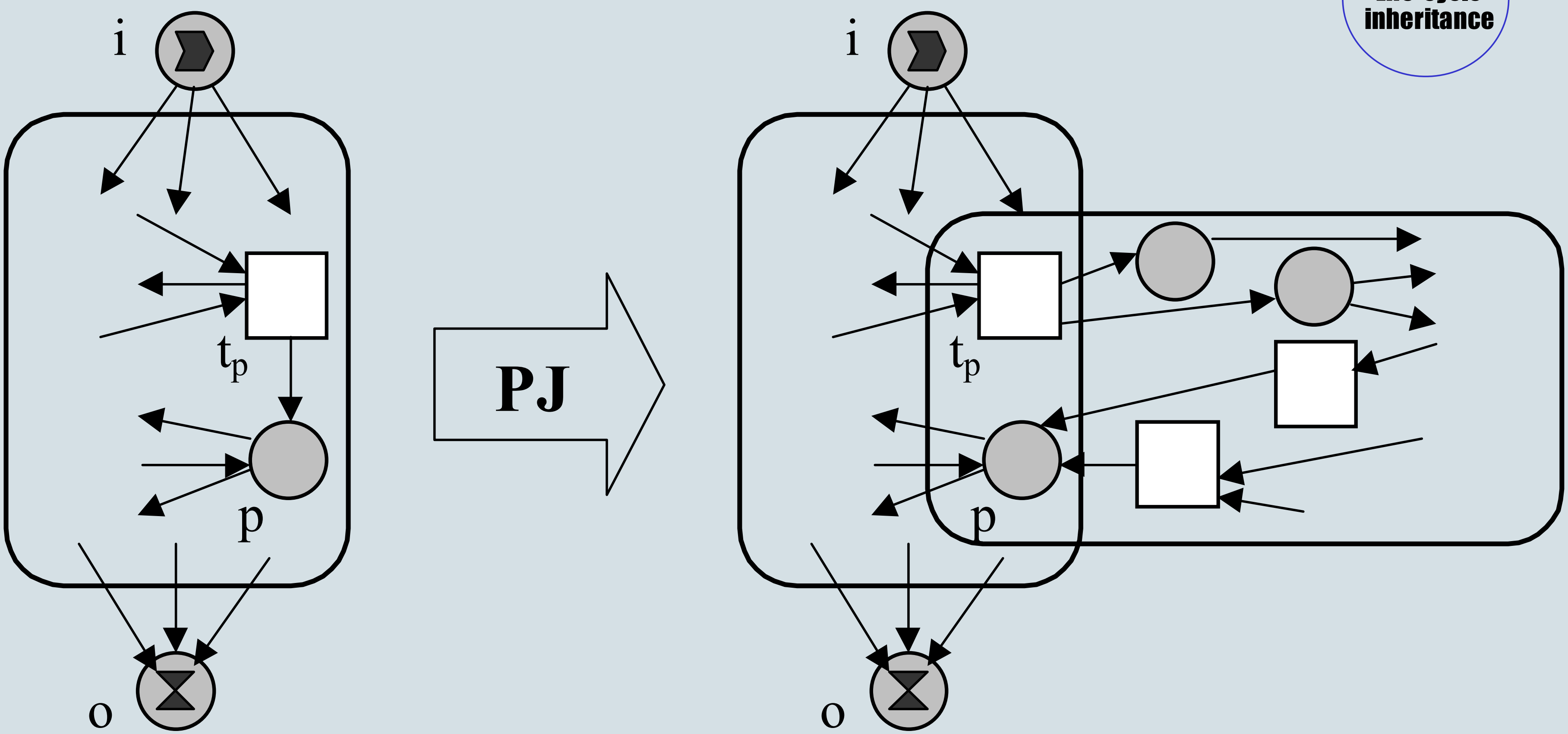
Transformation rule PT: adding alternatives





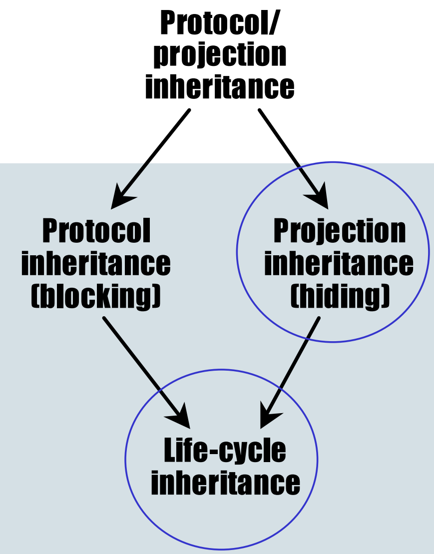
Transformation rule PJ: inserting steps

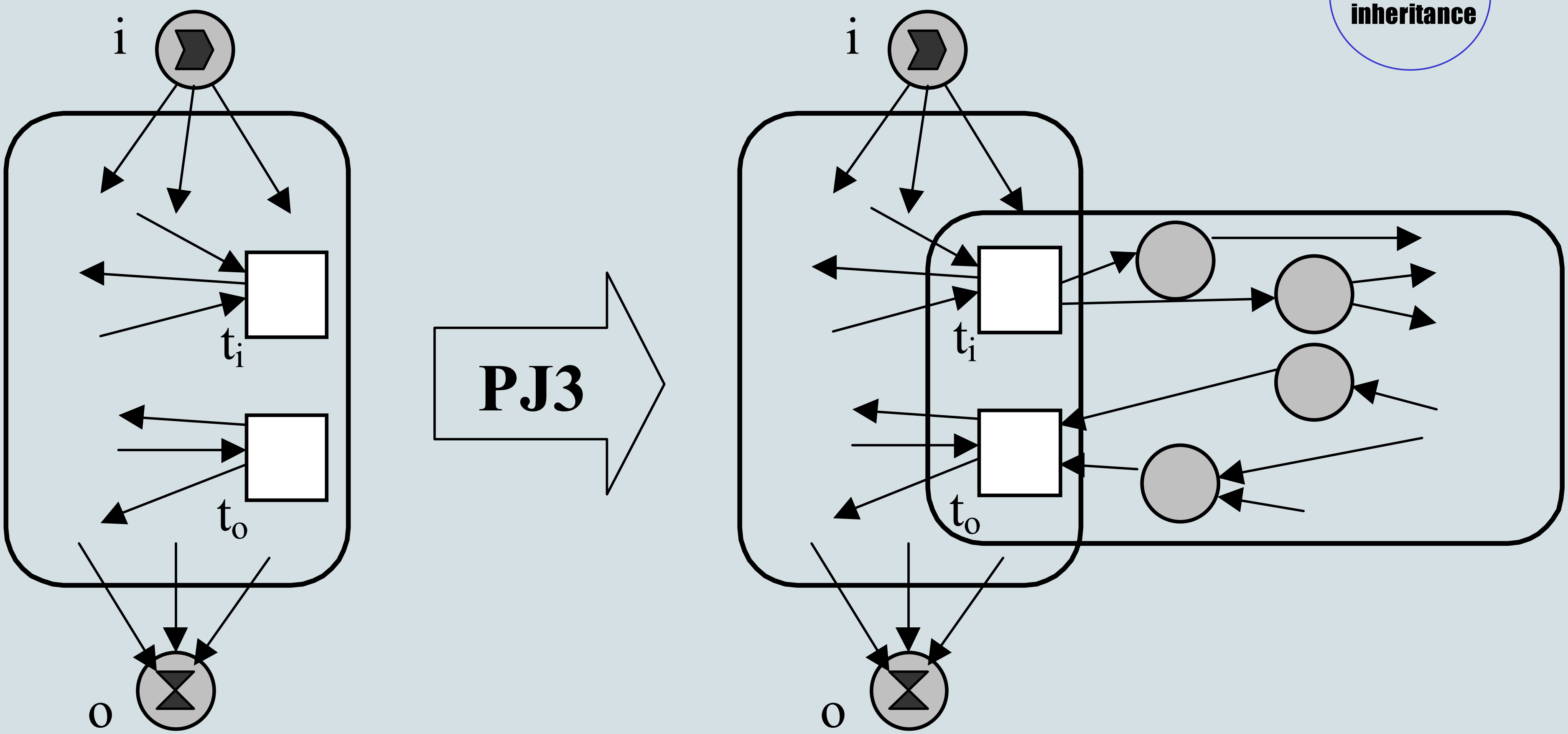




technische universiteit eindhoven

Transformation rule PJ3: add parallel behavior





Inheritance of behavior in UML

- The goal is to illustrate the four inheritance notions and the four transfer rules in the context of UML.
- The goal is NOT to provide a complete semantic mapping consistent with current standards.
- The four diagrams types that are relevant are:
  - Sequence diagrams
  - (Collaboration diagrams)
  - Statecharts diagrams
  - Activity diagrams

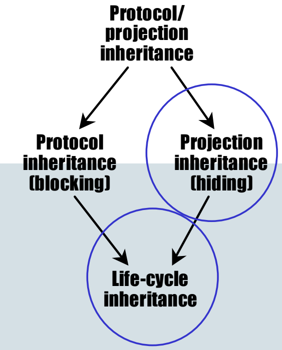
technische universiteit eindhoven

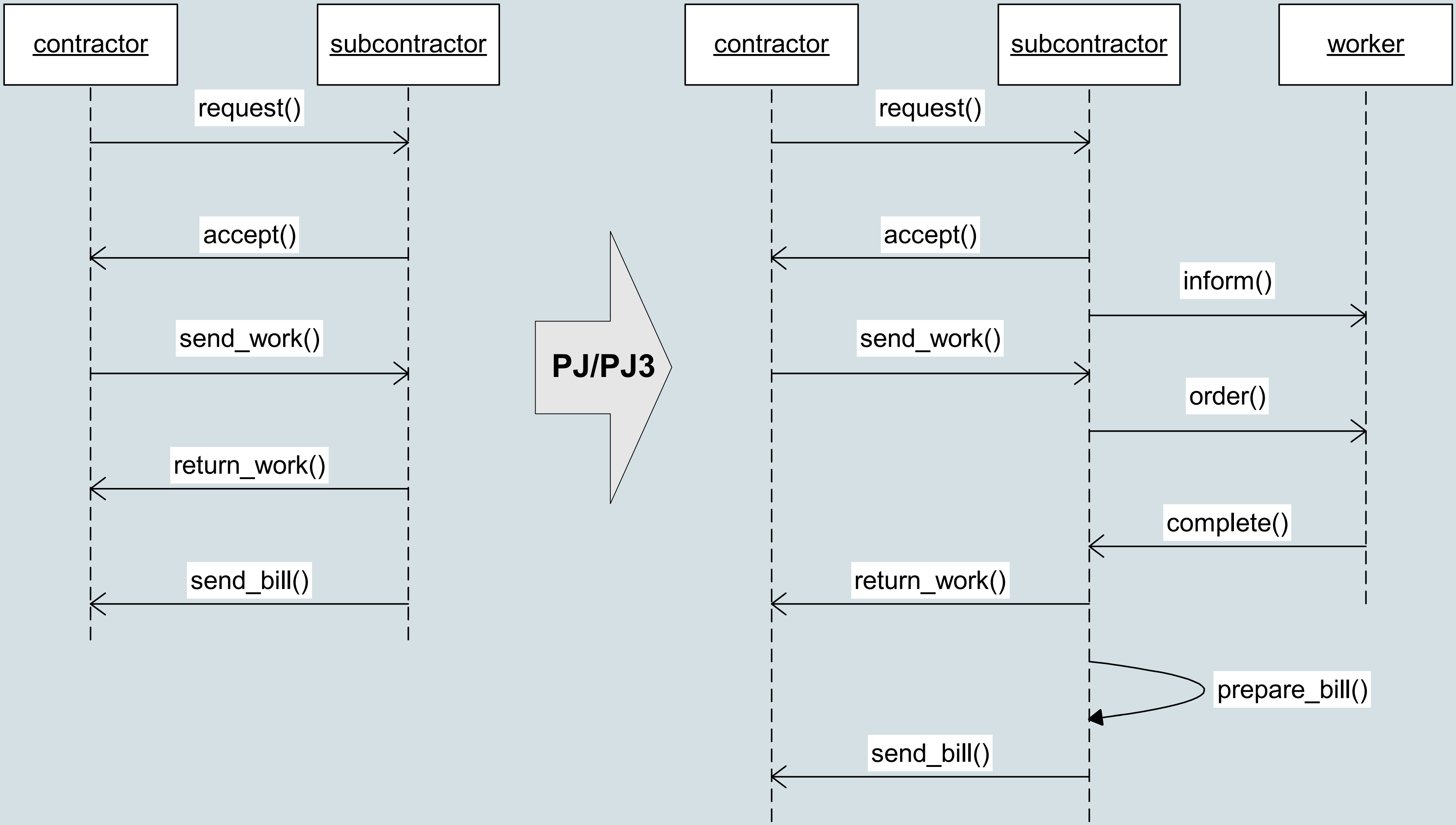
Sequence diagrams

- *Constructs considered:* lifelines, messages (communications of type procedure call, asynchronous and return), activation and concurrent branching.
- *Not considered:* more advanced constructs such as iteration, conditional and timed behavior.
- *Semantic domains*: TS and PN (marked graphs).
- *Relevant notions of inheritance:* projection inheritance.
- *Relevant transformation rules:* PJ, PJ3 and PP.

technische universiteit eindhoven

Example

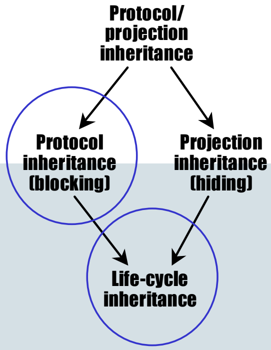




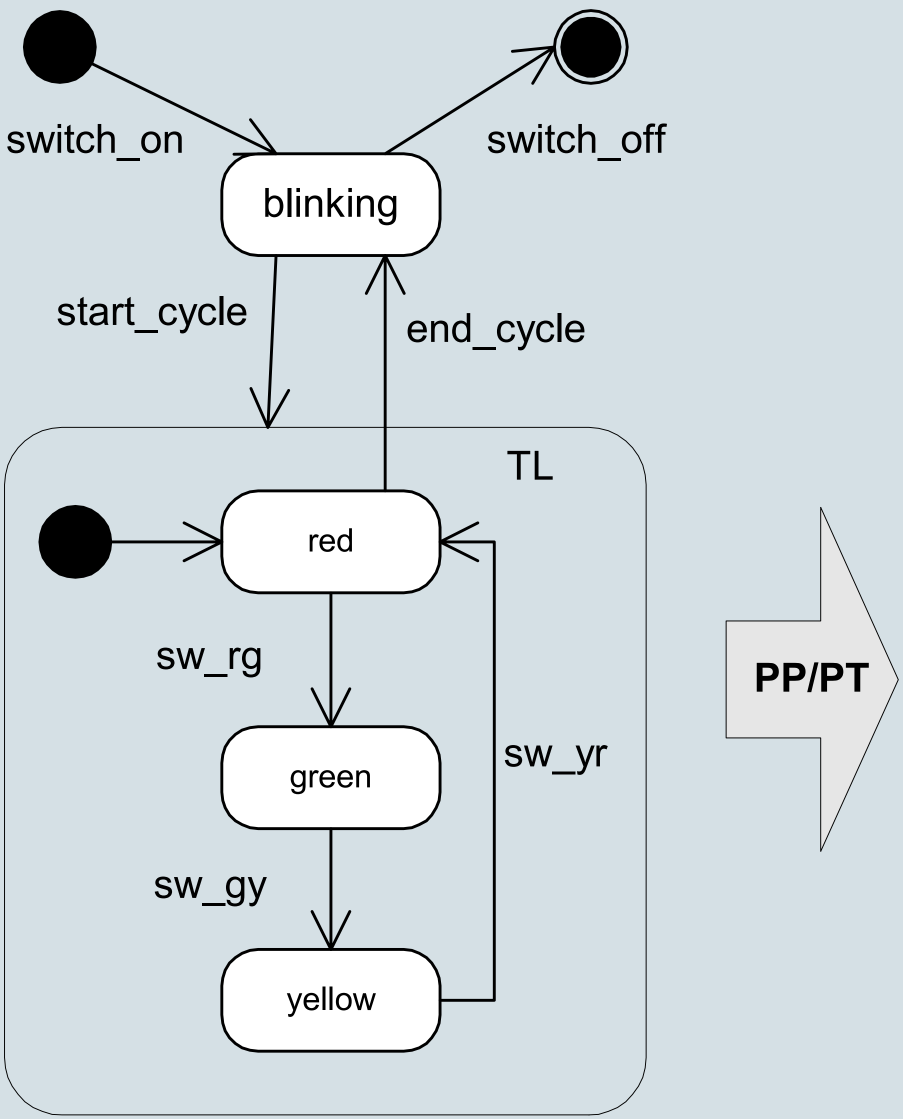
Statechart diagrams

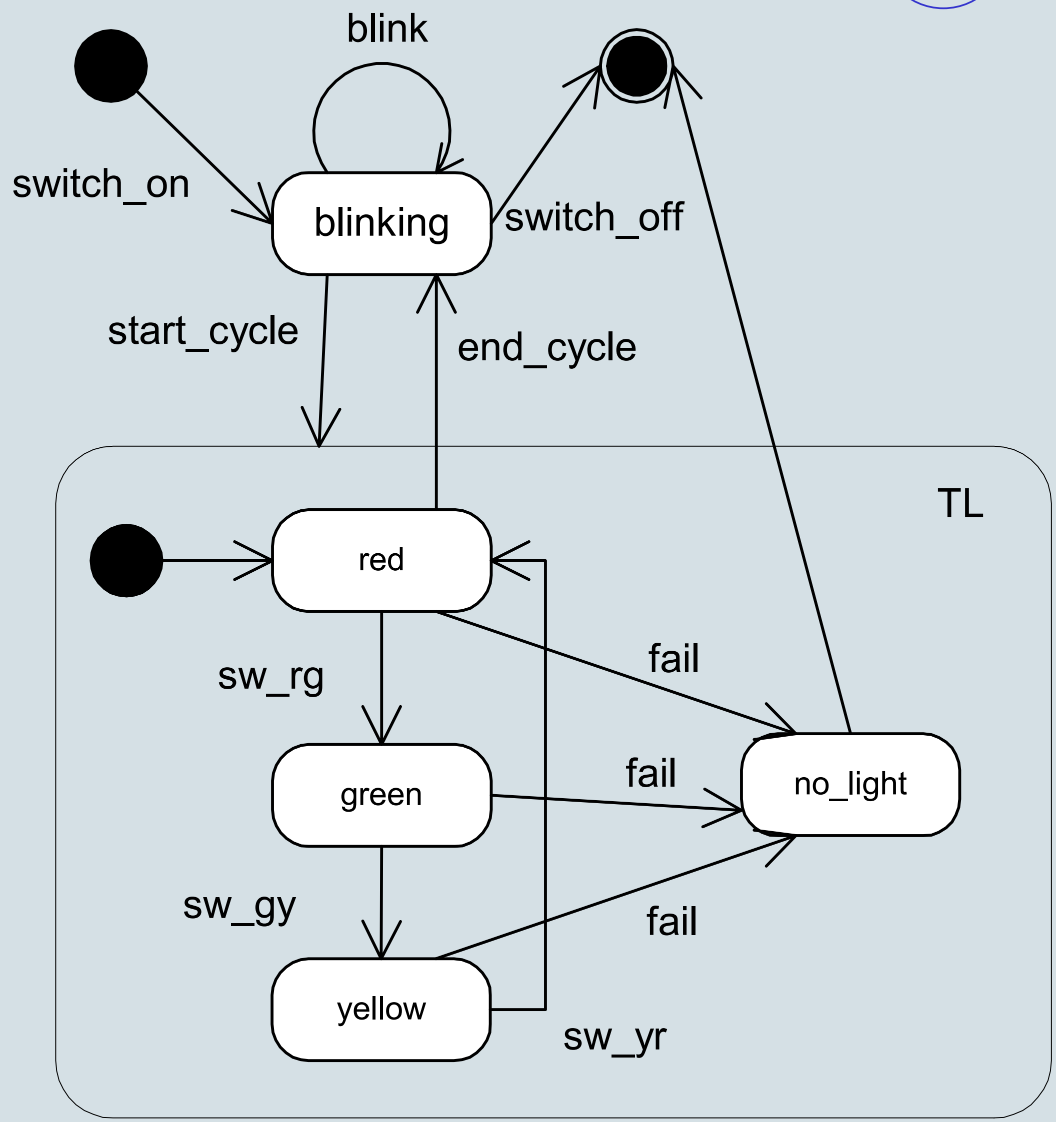
- *Constructs considered:* States, composite states, concurrent substates, transitions, compound transitions, etc.
- *Not considered:* data or time dependent behavior (e.g., abstraction from ECA rules).
- *Semantic domains*: TS, PA, and PN.
- *Relevant notions of inheritance:* all.
- *Relevant transformation rules:* all.

technische universiteit eindhoven

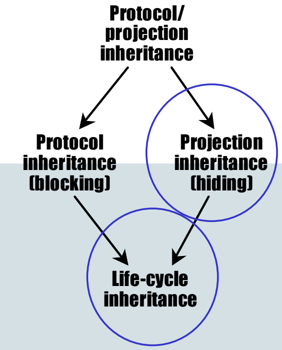


Example

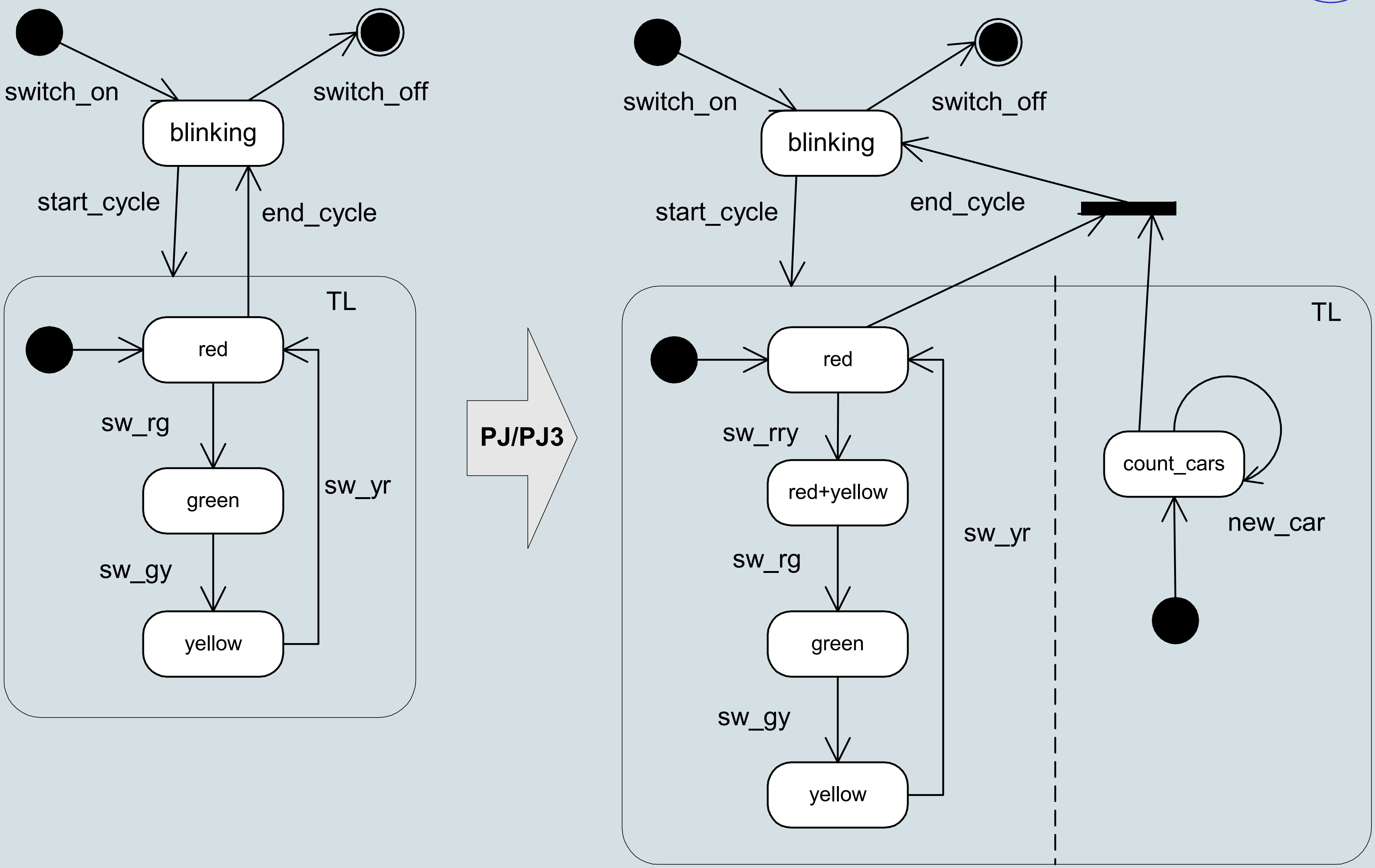




Example (2)



Protocol/ projection inheritance
Protocol inheritance (blocking)
Projection inheritance (hiding)
Life-cycle inheritance

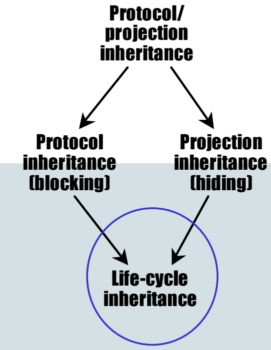


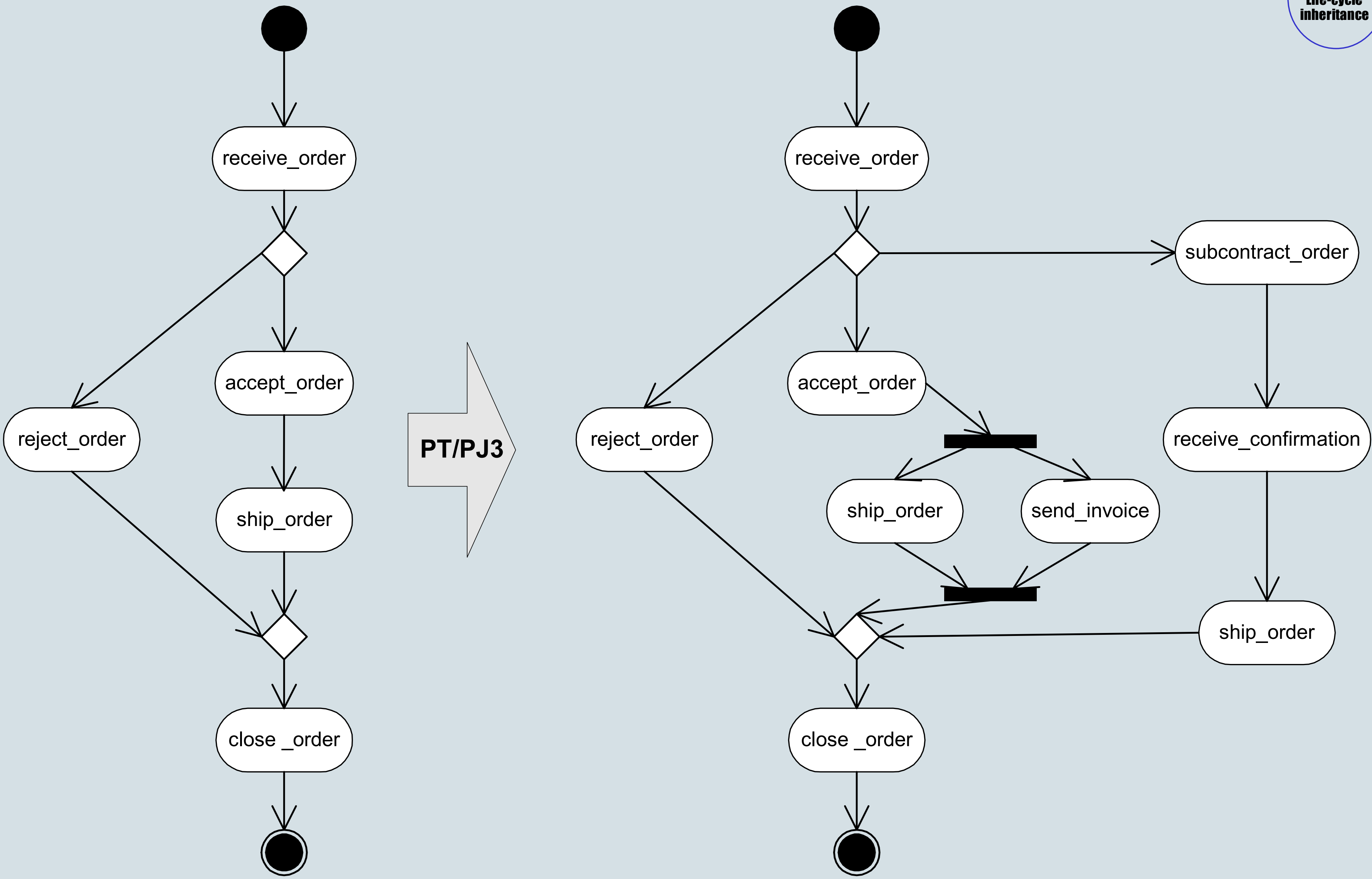
switch_on
switch_off
blinking
start_cycle
end_cycle
TL
red
sw_rg
green
sw_gy
yellow
sw_yr
PJ/PJ3
switch_on
switch_off
blinking
start_cycle
end_cycle
TL
red
sw_rry
red+yellow
sw_rg
green
sw_gy
yellow
sw_yr
count_cars
new_car

Activity diagrams

- *Constructs considered:* States, action states, decision/merge nodes, fork/join nodes, etc.
- *Not considered:* data or time dependent behavior (e.g., abstraction from ECA rules).
- *Semantic domains*: TS, PA, and PN.
- *Relevant notions of inheritance:* all.
- *Relevant transformation rules:* all.

Example


Protocol/ projection inheritance
Protocol inheritance (blocking)
Projection inheritance (hiding)
Life-cycle inheritance


receive_order
accept_order
reject_order
ship_order
close _order
PT/PJ3
receive_order
subcontract_order
accept_order
reject_order
receive_confirmation
ship_order
send_invoice
ship_order
close _order

Conclusion

- Four definitions of inheritance have been illustrated using the core semantic domain.
- Four transformation rules haven been illustrated using one of the intermediate semantic domains.
- To illustrate the applicability of these notions in the context of UML, examples have been given for sequence, statechart, and activity diagrams.

technische universiteit eindhoven

References

1. W.M.P. van der Aalst and T. Basten. Life-cycle Inheritance: A Petri-net-based Approach. In P. Az ´ ema and G. Balbo, editors, *Application and Theory of Petri Nets 1997*, volume 1248 of *Lecture Notes in Computer Science*, pages 62–81. Springer-Verlag, Berlin, 1997.
2. W.M.P. van der Aalst and T. Basten. Identifying Commonalities and Differences in Object Life Cycles using Behavioral Inheritance. In J.M. Colom and M. Koutny, editors, *Application and Theory of Petri Nets 2001*, volume 2075 of *Lecture Notes in Computer Science*, pages 32–52. Springer-Verlag, Berlin, 2001.
3. W.M.P. van der Aalst and T. Basten. Inheritance of Workflows: An Approach to Tackling Problems Related to Change. *Theoretical Computer Science*, 270(1-2):125–203, 2002.
4. P. America. Designing an Object-Oriented Programming Language with Behavioral Sub-typing. In J.W. de Bakker, W.P. de Roever, and G. Rozenberg, editors, *Foundation of Object-Oriented Languages*, volume 489 of *Lecture Notes in Computer Science*, pages 60–90. Springer-Verlag, Berlin, 1991.
5. T. Basten. *In Terms of Nets: System Design with Petri Nets and Process Algebra*. PhD thesis, Eindhoven University of Technology, Eindhoven, The Netherlands, December 1998.
6. T. Basten and W.M.P. van der Aalst. Inheritance of Behavior. *Journal of Logic and Algebraic Programming*, 47(2):47–145, 2001.

technische universiteit eindhoven

7. R. Breu, U. Hinkel, C. Hofmann, C. Klein, B. Paech, B. Rumpe, and V. Thurner. Towards a Precise Semantics for Object-Oriented Modeling Techniques. In M. Aksit and S. Matsuoka, editors, *11th European Conference on Object-Oriented Programming ECOOP'97*, volume 1241 of *Lecture Notes in Computer Science*, pages 344–366. Springer-Verlag, Berlin, 1997.

8. G. Engels, R. Heckel, and J.M. K¨ uster. Rule-Based Specification of Behavioral Consistency Based on the UML Meta-model. In M. Gogella and C. Kobryn, editors, *4th International Conference on The Unified Modeling Language (UML 2001)*, volume 2185 of *Lecture Notes in Computer Science*, pages 272–286. Springer-Verlag, Berlin, 2001.

9. R. Eshuis and R. Wieringa. Comparing Petri Nets and Activity Diagram Variants for Work-flow Modelling- A Quest for Reactive Petri Nets. In H. Ehrig, W. Reisig, and G. Rozenberg, editors, *Petri Net Technologies for Communication Based Systems*, Lecture Notes in Com-puter Science. Springer-Verlag, Berlin, 2002.

10. R.J. van Glabbeek. The Linear Time - Branching Time Spectrum II: The Semantics of Sequential Systems with Silent Moves. In E. Best, editor, *Proceedings of CONCUR 1993*, volume 715 of *Lecture Notes in Computer Science*, pages 66–81. Springer-Verlag, Berlin, 1993.

11. Object Management Group. *OMG Unified Modeling Language, Version 1.4*. OMG, http://www.omg.com/uml/, 2001.

12. Object Management Group. *OMG Unified Modeling Language 2.0 Proposal, Re-vised submission to OMG RFPs ad/00-09-01 and ad/00-09-02, Version 0.67*1. OMG, http://www.omg.com/uml/, 2002.
13. D. Harel. Statecharts: A Visual Formalism for Complex Systems. *Science of Computer Programming*, 8:231–274, 1987.
14. D. Harel and O. Kupferman. On the Behavioral Inheritance of State-Based Objects. Techni-cal Report MCS99-12, The Weizmann Institute of Science, Israel, 1999.
15. G. Kappel and M. Schrefl. Inheritance of Object Behavior - Consistent Extension of Object Life Cycles. In J. Eder and L.A. Kalinichenko, editors, *Proceedings of the Second Inter-national East/West Database Workshop (EWDW 1994)* , pages 289–300. Springer-Verlag, Berlin, 1995.
16. E. Kindler, A. Martens, and W. Reisig. Inter-Operability of Workflow Applications: Local Criteria for Global Soundness. In W.M.P. van der Aalst, J. Desel, and A. Oberweis, editors, *Business Process Management: Models, Techniques, and Empirical Studie*s, volume 1806 of *Lecture Notes in Computer Scienc*e, pages 235–253. Springer-Verlag, Berlin, 2000.
17. B. Liskov and J. Wing. A Behavioral Notion of Subtyping. *ACM Transactions on Program-ming Languages and System*s, 16(6):1811–1841, November 1994.
18. O. Nierstrasz. Regular Types for Active Objects. *ACM Sigplan Notice*s, 28(10):1–15, Octo-ber 1993. Special issue containing the proceedings of the 8th. annual conference on Object-Oriented Programming Systems, Languages, and Applications, OOPSLA'93, Washington DC, 1993.

20. E. Rudolph, J. Grabowski, and P. Graubmann. Tutorial on Message Sequence Charts. *Com-puter Networks and ISDN Systems*, 28(12):1629–1641, 1996.
21. J. Rumbaugh, I. Jacobson, and G. Booch. *The Unified Modeling Language Reference Man-ual*. Addison Wesley, Reading, MA, USA, 1998.
22. M. Schrefl and M. Stumptner. On the Design of Behavior Consistent Specialization of Object Life Cycles in OBD and UML. In M. Papazoglou, S. Spaccapietra, and Z. Tari, editors, *Advances in Object-Oriented Data Modelling*, pages 65–104. MIT Press, 2000.
23. M. Stumptner and M. Schrefl. Behavior Consistent Inheritance in UML. In Alberto H. F. Laender et al. editor, *Proceedings of the 19th International Conference on Conceptual Modeling (ER 2000)*, volume 1920 of *Lecture Notes in Computer Science*, pages 527–542. Springer-Verlag, Berlin, 2000.
24. 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.
25. H. Wehrheim. Subtyping Patterns for Active Objects. In H. Giese and S. Philippi, editors, *Proceedings 8ter Workshop des GI Arbeitskreises GROOM (Grundlagen objekt-orientierter Modellierung)*, volume 24/00, M¨ unster, Germany, 2000. University of M¨ unster.
26. R.J. Wieringa. *Algebraic Foundations for Dynamic Conceptual Models*. PhD thesis, Free University, Amsterdam, The Netherlands, 1990.

technische universiteit eindhoven

BPM 2003

INTERNATIONAL CONFERENCE ON BUSINESS PROCESS MANAGEMENT

On the Application of Formal Methods to "Process-Aware" Information Systems

Eindhoven, The Netherlands, June 26-27, 2003

http://www.tm.tue.nl/it/bpm2003