ISYM - Information Systems & Management

Prof. Dr. Peter Loos

Paper 17

Peter Fettke

# Overview of the Unified Modeling Language – Extension of an Article

2004

Working Papers of the Research Group Information Systems & Management

Publisher:

Prof. Dr. Peter Loos
Johannes Gutenberg-University Mainz
ISYM - Information Systems & Management
Lehrstuhl für Wirtschaftsinformatik und BWL
D-55099 Mainz, Germany

http://www.isym.bwl.uni-mainz.de

The working papers 1 through 8 are published in Chemnitz, Germany, by Prof. Dr. Peter Loos and Prof. Dr. Bernd Stöckert.

ISSN 1617-6324 (printed version)
ISSN 1617-6332 (Internet version)

URN urn:nbn:de:0006-0175

## Management Summary

Mature engineering disciplines are generally characterized by accepted methodical standards for describing all relevant artifacts of their subject matter. Such standards not only enable practitioners to collaborate, but they also contribute to the development of the whole discipline. In 1994, Grady Booch, Jim Rumbaugh, and Ivar Jacobson joined together to unify the plethora of existing object-oriented systems engineering approaches at semantic and notation level. Their effort leads to the Unified Modeling Language (UML), a well-known, general-purpose, tool-supported, process-independent, and industry-standardized modeling language for visualizing, describing, specifying, and documenting systems artifacts. This article overviews UML's main concepts and gives some insights into advanced topics. Furthermore, the future of UML is discussed. The discussion is based on UML version 1.5.

**Keywords:** conceptual modeling, information modeling, object-oriented modeling, model, object-oriented analysis & design, meta-model, methodology

This paper is an extension of the article "Unified Modeling Langauge" published in the "Encyclopedia of Information Science and Technology (5 Volumes)", edited by Mehdi Khosrow-Pour.

## Author

*Peter Fettke*
Johannes Gutenberg-University Mainz
ISYM - Information Systems & Management
Lehrstuhl für Wirtschaftsinformatik und BWL
D-55099 Mainz, Germany
Phone: +49 6131 39-22734, Fax: -22185
E-Mail: fettke@isym.bwl.uni-mainz.de

# Contents

# 1 Introduction

Mature engineering disciplines are generally characterized by accepted methodical standards for describing all relevant artifacts of their subject matter. Such standards not only enable practitioners to collaborate, but they also contribute to the development of the whole discipline. In the early 1990s, there was a multitude of different object-oriented system engineering methods, for instance Booch, 1994; Coad & Yourdon, 1991; Jacobson, Christerson, Jonsson, & Övergaard, 1992; Martin & Odell, 1998; Rumbaugh, Blaha, Premerlani, Eddy, & Lorensen, 1991; Shlaer & Mellor, 1988; Wirfs-Brock, Wilkerson, & Wiener, 1990. All these approaches are kind of similar, although there are subtle differences in semantics and often major differences in methodology and notation.

In 1994, Grady Booch, Jim Rumbaugh, and Ivar Jacobson (later) joined together to unify their approaches at semantic and notation level (Booch, 1999, 2002; Fowler, 2004; OMG, 2003d; Rumbaugh, Jacobson, & Booch, 1998). Their effort leads to the Unified Modeling Language (UML), a well-known, general-purpose, tool-supported, process-independent, and industry-standardized modeling language for visualizing, describing, specifying, and documenting systems artifacts. Table 1 depicts the origin and descent of UML.

| **Version** | **Year** | **Comments** |
|---|---|---|
| 0.8 | 1995 | Origin of UML, so-called “Unified Method” |
| 0.9 | 1996 | Refined proposal |
| 1.0 | 1997 | Initial submission to OMG |
| 1.1 | 1997 | Final submission to OMG |
| 1.2 | 1998 | Editorial revision with no significant technical changes |
| 1.3 | 1999 | New use case relationships, revised activity diagram semantics |
| 1.4 | 2001 | Minor revisions, addition of profiles |
| 1.5 | 2003 | Adding action semantics |
| 2.0 | 2004 (?) | Planned major revision, deep changes to meta-model, new diagram types |

**Table 1: History of UML (Fowler, 2004, p. 151-159; Kobryn, 1999, p. 30)**

UML is applicable to software and non-software domains, including software architecture (Medvidovic, Rosenblum, Redmiles, & Robbins, 2002), real-time and embedded systems (Douglass, 1998), business applications (Eriksson & Penker, 2000), manufacturing systems (Bruccoleri, Dieaga, & Perrone, 2003), electronic commerce systems (Saleh, 2002), data warehousing (Dolk, 2000), bioinformatics (Bornberg-Bauer & Paton, 2002) and others. The language uses multiple views to specify system’s structure and behavior. The recent version UML 1.5 supports nine different diagram types, namely: class, object, use case, sequence, collaboration, statechart, activity, component, and deployment diagrams. Table 2 and Appendix A overview the main concepts of each diagram, a more detailed description is given below. For a full discussion of all semantics see Booch, Rumbaugh, & Jacobson, 1999; D'Souza & Wills, 1998; Fowler, 2004; Rumbaugh et al., 1998; Schader & Korthaus, 1998; Siau & Halpin, 2001.

| Focus | Diagram | Purpose | Main Concepts |
|---|---|---|---|
| Static diagrams | Class | Object structure | Class, features, relationships |
| | Object | Example configuration of instances | Object, link |
| Dynamic diagrams | Use case | User interaction with system | Use case, actor |
| | Sequence | Interaction between objects emphasizing sequences | Interaction, message |
| | Collaboration | Interaction between objects emphasizing collaborations | Collaboration, interaction, message |
| | Statechart | Change of events during object's lifetime | State, transition, event, action |
| | Activity | Procedural and parallel behavior | State, activity, completion, transition, fork, join |
| Implementation diagrams | Component | Structure and connections of components | Component, interface, dependency |
| | Deployment | Deployment of components to nodes | Node, component, dependency |

**Table 2: UML Diagram Types**

The specification of the UML is publicity available and maintained by the Object Management Group (OMG), a non-profit organization founded in 1989 with the objective to archive systems' interoperability. OMG's standardization process is formalized and consists of several proposal, revision, and final implementation activities (Kobryn, 1999, p. 31f.). Modeling tools supporting the constructing, verification, and maintenance of UML diagrams are available from a number of commercial vendors and the open source community (OMG, 2004c; Robbins & Redmiles, 2000).

# 2 Background

There is a great deal of terminological confusion in the modeling literature. For example, the term "model" is often used for different purposes. A modeling language or grammar provides a set of constructs and rules that specify how to combine the constructs to model a system (Wand & Weber, 2002, p. 364). Furthermore, it can be distinguished between an abstract syntax and a concrete syntax or notation of a language. While the abstract syntax specifies conceptual relationships between the constructs of the language, the concrete notation defines symbols representing the abstract constructs. In contrast, a modeling method provides procedures by which a language can be used. A consistent and suited set of modeling methods is called a methodology. A model is a description of a domain using a particular modeling language.

The UML specification provides an abstract syntax and a concrete notation for all UML diagrams as well as an informal description of the constructs' semantics. The UML's language specification is independent of but strongly related to other OMG standards such as Common Data Warehouse Model (OMG, 2003a), XML Metadata Interchange (OMG, 2003h) or Meta Object Facility (OMG, 2002). A modeling method or a modeling methodology is not defined by the UML standard. Therefore, the language is process-neutral and can be used with different software development processes such as Unified Software Development Process (Jacobson, Booch, & Rumbaugh, 1998), the Personal Software Process (Humphrey, 1995), or even agile programming techniques, e.g. eXtreme Programming (Beck, 2000).

Although (conceptual) modeling has a long history (Chen, 1976; Mylopoulos, 1998) and a wide variety of different modeling approaches exist in literature (Hofstede & Weide, 1993), no other modeling language gains so much attention in software industry. Other modeling approaches that are to a certain degree accepted in practice, for instance the Entity-Relationship Model or flow charts, have a much more limited scope than UML. These approaches address just some aspects of systems' specification, namely the data resp. process view. In contrast, UML supports the specification of static as well as dynamic aspects. Other approaches with a similar scope, for instance Open Modeling Language (Firesmith, Henderson-Sellers, & Graham, 1998) or Object-oriented Systems Modeling (Jackson, Liddle, & Woodfield, 1998), are not widely accepted in practice.

# 3 Main Concepts

## 3.1 Structural Diagrams

Structural or static diagrams describe the objects of a system in terms of classes, attributes, operations, relationships, and interfaces. This type of diagrams includes (1) class and (2) object diagrams.

(1) *Class diagram*. A class diagram can be viewed as a graph of several elements connected by static relationships. The main element is a class. Classes represent concepts within the system being modeled and are descriptors for a set of objects with similar structure, behavior, and relationships. An object represents a particular instance of a class. Each class has a unique name among other classes within a specific scope (usually a UML package). A class can hold several attributes and operations. Attributes have names and belong to particular types that can be simple data types such as Integer, String, Boolean as well as complex types (e.g. other classes). Operations are services offered by an instance of the class and may be requested by other objects during run-time.

There are three kinds of relationships between classes: (i) Associations describe properties that do not belong to one but to two or more objects. An association can be specified by a multiplicity that describes how many objects of each class can participate in that association. Furthermore, it can be distinguished between unary, binary, ternary or n-nary associations regarding how many classes are participating in that association. (ii) Aggregation or composition describe whole-part relationships between classes. There is a strong discussion about precise semantics of this kind of relationship (Barbier, Henderson-Sellers, Parc-Lacayrelle, & Bruel, 2003). (iii) To avoid redundant class specifications, generalization relationships between classes can be defined. A generalization is a relationship between a more general class (super-class or parent) and a more specific class (sub-class or child) that is fully consistent with the super-class and adds additional information.

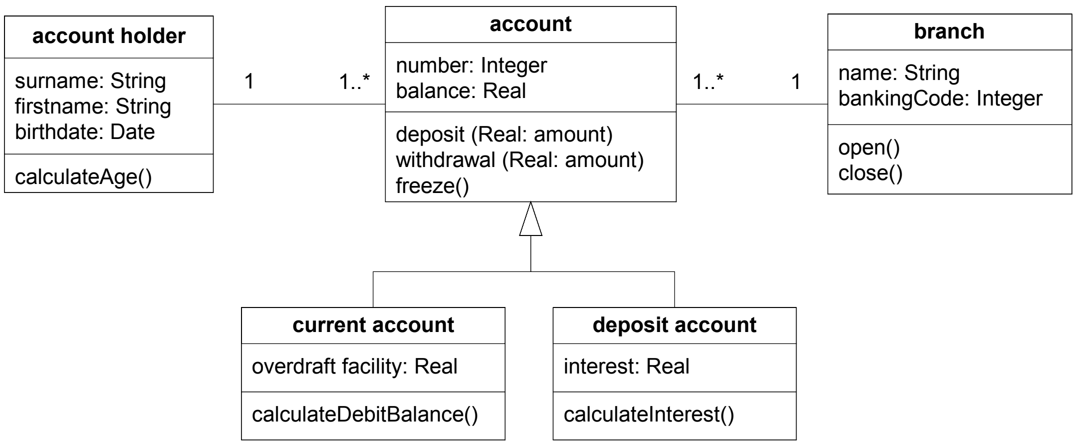


**Figure 1: Class diagram for banking systems**

Figure 1 depicts a class diagram for banking systems. An account is described by the attributes ‘number’ and ‘balance’. The operations ‘deposit’, ‘withdrawal’, and ‘freeze’ are offered by an account. Each account is kept by a ‘branch’ and is assigned to a ‘holder’. The classes ‘deposit account’ and ‘current account’ reuse the structure and behavior of the class ‘account’ (inheritance relationship). In addition, the specialized account classes define further feature, e.g. an object of the class ‘current account’ is described by the property ‘overdraft facility’ and offers an operation calculating the current debit balance.

(2) *Object diagram*. An object diagram is an instance of a class diagram and depicts the state of the system at a point in time. It contains objects including their actual values of attributes and links describing object references. The application range of object diagrams is limited compared to other diagrams. However, it is useful to show some examples of data structures and a particular configuration of several objects.

## 3.2 Behavioral Diagrams

Behavioral diagrams describe the dynamics between objects of a system in terms of interactions, collaborations, and state histories. This type of diagram includes (1) use case, (2) sequence, (3) collaboration (4) state chart and (5) activity diagrams.

(1) *Use case diagram*. The idea “use case” was introduced by Jacobson, 1987 and resembles other concepts such as scenarios or scripts (Rumbaugh et al., 1991). A use case specifies a complete set of events within a system to fulfill tasks or transactions in an application from a user’s point of view. In a use case diagram, a set of use cases, actors, and relationships between these elements are depicted. Several use cases may optionally be enclosed by a rectangle that represents the boundary of the containing system. An actor describes a particular role of a human or non-human user of the system being modeled. In practice, there is no consensus on the level of abstraction writing effective use cases

(Dobing & Parsons, 2000) and each use case is often enriched with textual information (Schneider & Winters, 2001).

(2) *Sequence diagram*. Sequence diagrams describe interactions between different objects. An interaction consists of a partially ordered set of messages that are exchanged by the participants of that interaction. Sequence diagrams have two dimensions: The horizontal dimension represents the participants of the interaction; the vertical dimension represents the flow of time (usually time proceeds from up to down). In real-time systems, time is not conceptualized by a set of instants but a time metric.

(3) *Collaboration diagram*. Collaboration and sequence diagrams use the same underlying information and can easily be transformed into each other. While sequence diagrams emphasize the sequence of communication between objects, collaboration diagrams show the roles of the participants of an interaction and their relationships. A sequence number specifies the flow of messages in an interaction, so no time dimension is needed in this diagram. Simple communication patterns can be depicted by collaboration diagrams; sequence diagram can better specify complex message exchanges or requirements for real-time systems.

(4) *Statechart diagram*. Object behavior is represented by statechart diagrams which are primary based on Harel's work on visual machines (Harel, 1987) and resembles concepts of traditional finite state machines. This diagram can be used both to specify an entire object or a single method. A state describes a condition during the lifetime of an object. Transitions are relationships between two states describing that an object's state can change from the first to the second state. The change of a state is triggered by an event that occurs in the modeled system. There are two special types of states: An initial state identifies the point at which behavior starts when an object is created, a final state identifies the point at which behavior ends (end of object's lifetime).

(5) *Activity diagram*. While statechart diagrams are used to specify the behavior of a single object, activity diagrams can describe behavior that crosses object boundaries. They are analogous to traditional flowcharts and are often used to document (business) processes or the dynamics inside a use case. So-called fork bars are used to describe activities that can be executed in parallel. Parallel activities get synchronized by so-called join bars. Guards are used to specify conditional forks that are only executed if particular conditions hold.

## 3.3 Implementation Diagrams

Implementation diagrams capture the physical structure of a software system during build- and runtime.

(1) *Component diagram*. Components in UML are physical elements such as source, binary or executable modules resp. files. Simple components can be aggregated to complex components to specify physical containment relations. Directed relationships between components specify that one component relies or refines the other. Such relationships are called dependencies.

(2) *Deployment diagram*. While component diagrams primary show build-time dependencies of components, deployment diagrams show a run-time configuration of the system's components. In addition,

a deployment diagram uses nodes representing a processing resource, for instance a server or workstation, that can execute system operations during run-time.

All described structural, behavioral and implementation diagrams can capture many other types of semantics, such as parameterized classes, association classes, abstract classes, interfaces, composite objects, notes, association roles, packages, polymorphic operations, use case associations, composite states, hierarchical states, composite triggers, guards etc. (Booch et al., 1999; Fowler, 2004; OMG, 2003d; Rumbaugh et al., 1998).

## 4 Advances Topics

*Object Constraint Language (OCL)*. A UML diagram is typically not refine enough to capture unambiguously all relevant aspects of a system's specification. There is often a need to describe further necessary constraints in a more precise and rigorous way. OCL is used for that purpose. It is a formal textual language inspired by the 'Design by Contract' concept (Meyer, 1997) and provides concepts for the definition of constraints such as invariants, pre- and post-conditions (Cengarle & Knapp, 2004; Warmer & Kleppe, 2003).

*Language specification and meta-model*. The UML itself is specified using textual descriptions and a four-layered meta-modeling approach (Atkinson & Kühne, 2002, pp. 291-296; Kobryn, 1999, 32-34). In this approach, the semantic constructs at each layer are recursively refined. The top layer, the meta-meta-model (M3), provides a so-called Meta Object Facility (MOF, (OMG, 2002)) to specify meta-models on the next lower layer. The MOF is used on the meta-model (M2) layer to specify the concepts of UML diagrams, e.g. class diagram etc. The model (M1) and object (M0) layer are user-defined. The former specifies concrete UML models, the later instances of the former. Note, that the meta-model approach addresses UML's abstract syntax, the semantic is not yet fully formalized.

*Extension mechanisms*. The UML is designed as a universal language. However, it may be necessary to adopt the UML to problem-domain specific needs. There are heavyweight and lightweight extension mechanisms (OMG, 1999). Heavyweight extensions are supported by MOF and carried out on the meta-model (M2) layer. Such extensions have great impact on the language and are not performed by a particular modeler. User extensions are usually lightweight extensions that are built-in mechanisms of the UML. Lightweight extensions comprise constraints (OCL expressions), tagged values (attached additional information to model elements), and stereotypes (most powerful lightweight mechanism ranging from concrete syntax modifications to semantics redefinitions (Berner, Glinz, & Joos, 1999)).

## 5 Future Trends

The forthcoming UML Version 2.0 (UML 2) was at first planned for 2001 (Kobryn, 1999, p. 30) but is until now (early 2004) not fully completed. In the meantime a strong discussion about what UML 2 should and should not be evolved (Dori, 2002; Duddy, 2002; Engels, Heckel, & Sauer, 2001; W. Frank & Tyson, 2002; Kobryn, 2002; Mellor, 2002; Miller, 2002; Selic, Ramackers, & Kobryn, 2002). Currently, the UML 2 standard is voted to recommend by the OMG's technical board (OMG, 2003e) and is in its finalization phase. It consists of four separate documents (Kobryn, 2002). The UML 2 Infrastructure Specification is concerned with core language features (OMG, 2003f). Advanced topics such as component and activity modeling are specified in the UML Superstructure Specification (OMG, 2003g). The OCL and the Diagram Interchange Specification are two further separate UML 2 specification documents (OMG, 2003c). This major revision mainly focuses on language extensibility, language specification, language precision and expressiveness. Although the complete language specification is almost fully rewritten, this revision will be primary an internal reorganization with just minor consequences for the end user. For example, the new proposed diagrams mainly clarify and resemble existing diagram types.

Further trends include:

(1) *Model Driven Architecture* (MDA): MDA promotes modeling through the whole system's life-cycle (Bock, 2003; Fettke & Loos, 2003b; Frankel, 2003; OMG, 2003b; Selonen, Koskimies, & Sakkinen, 2003). Its objective is to fully automate the system's development process.

(2) *Executable UML*: Executable UML enriches the modeling concepts with execution semantics (Mellor & Balcer, 2002). This opens the possibility of software development without "classical" programming.

(3) *Model libraries*: UML is used to standardize domain specific models fostering the (re-)use of reference models (Fettke & Loos, 2003a), e.g. OMG's Business Enterprise Integration (OMG, 2004a) or Finance Domain Task Forces (OMG, 2004b).

(4) *Ontological analysis and semantics*: This research line evaluates UML from an ontological point of view and incorporates real-world semantics to UML constructs (Evermann & Wand, 2001; Opdahl & Henderson-Sellers, 2002). The aim of an ontological evaluation is to examine if all constructs of an ontology can be mapped onto the constructs of UML and vice versa.

(5) *Component-based development*: UML is primary an object-oriented language. To fully support component-based development, some enhancements are needed (Dahanayake, 2003; Fettke & Loos, 2003c; Kobryn, 2000). Particularly, component descriptions must include dependencies on other components, quality specifications for needed and offered services, and domain-specific semantics.

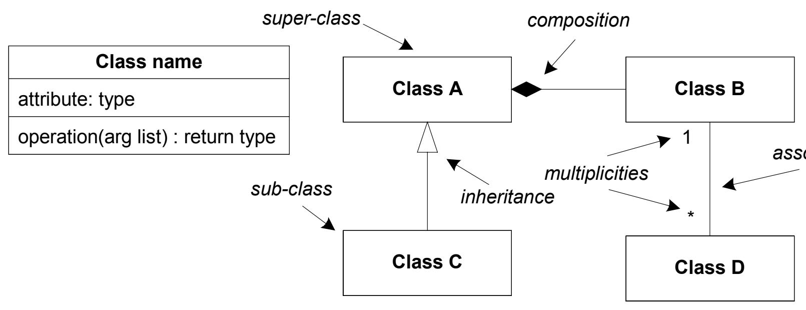
# 6 Conclusion

Although almost everyone acknowledges the practical benefits of a standardized modeling language (e.g. protection of investments in technology, easier model exchange and reuse, better professional training (U. Frank, 1997, p. 13)), there are important opportunities that have to be challenged. UML's size (UML 2 has approximately 1000+ pages) and complexity is compared with other languages overwhelming (Siau & Cao, 2001). Therefore users have difficulties in writing and reading diagrams (Agarwal & Sinha, 2003; Laitenberger, Atkinson, Schlich, & Emam, 2000) and tool venders have problems to fully support the UML standard. Furthermore, the maintenance of the standard is very expensive and error-prone, e.g. Fuentes, Quintana, Llorens, Génova, & Prieto-Díaz, 2003 identified several hundred errors in the UML meta-model. Other authors criticize UML for its semantic inconsistency, construct ambiguity, notation inadequacy, and cognitive misdirection (Champeaux, 2003; U. Frank, 1998; Henderson-Sellers, 2002; Kobryn, 2004; McLeod, Halpin, Kangassalo, & Siau, 2001; Shen & Siau, 2003; Thomas, 2002; Wang, 2001).

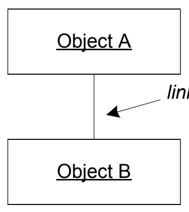
On the other hand, UML is the de-facto standard for object-oriented modeling and an important milestone in software engineering. Modeling of software systems increases the degree of abstraction during system development tremendously. This change is similar to the replacement of assembly languages by high-level languages in the 1960s and 1970s. Today, high level languages are not used in all but most domains. We predict that, in the future, UML has an analogous position as high-level languages have today. Therefore, UML continues to play a major role in systems development.

# Appendix A: UML Diagram Examples

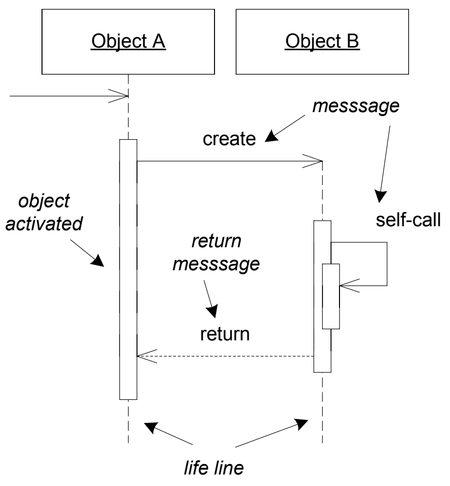
**Class diagram**



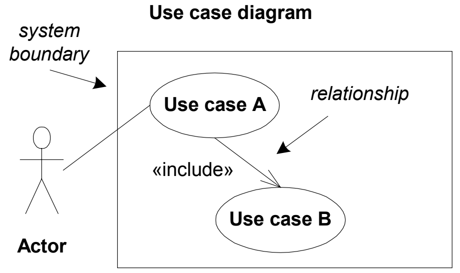
**Object diagram**



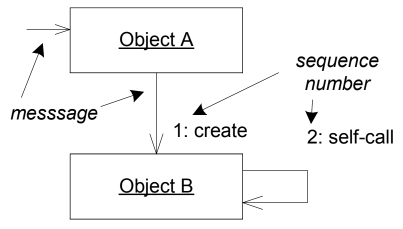
**Sequence diagram**



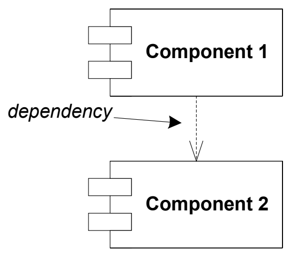
**Use case diagram**



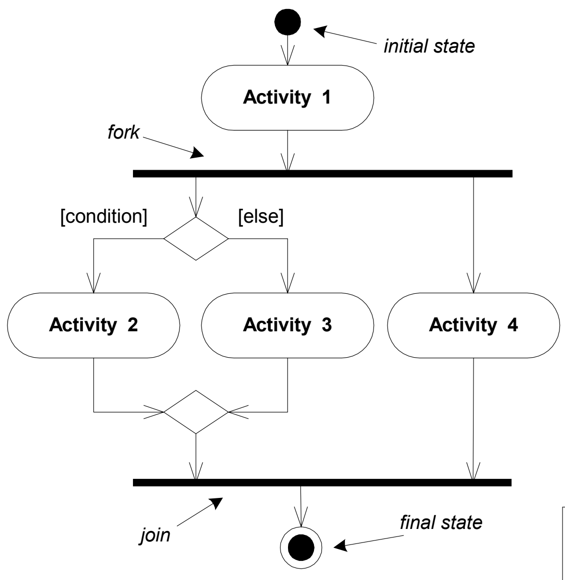
**Collaboration diagram**



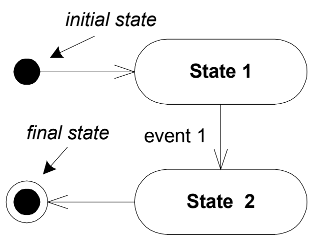
**Component diagram**



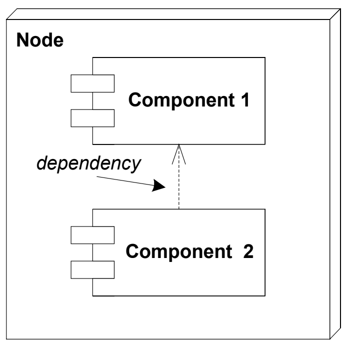
**Activity diagram**



**Statechart diagram**



**Deployment diagram**



**Legend:**
**Diagram type**
*Explanatory note* ⟶

## Appendix B: Conferences

The Unified Modeling Language «UML», annual, since 1998,
URL: http://dblp.uni-trier.de/db/conf/uml/index.html

International Conference on Conceptual Modeling (ER), annual, since 1979,
URL: http://conceptualmodeling.org/

Conference on Object-Oriented Programming Systems, Languages, and Applications (OOPSLA), annual, since 1986,
URL: http://www.oopsla.org/

European Conference on Object-Oriented Programming (ECOOP), annual, since 1987,
URL: http://www.ecoop.org/

## Appendix C: Online Resources

The UML Bibliography,
URL: http://dustbin.informatik.uni-bremen.de/umlbib/

Object Management Group's UML Resource Page,
URL: http://www.omg.org/uml/

Rational's UML Resource Center,
URL: http://www.ibm.com/software/rational/uml/

## References

Agarwal, R., & Sinha, A. P. (2003). Object-Oriented Modeling with UML: A Study of Developers' Perceptions. *Communications of the ACM, 46*(9), 248-256.

Atkinson, C., & Kühne, T. (2002). Rearchitecting the UML Infrastructure. *ACM Transactions on Modeling and Computer Simulation, 12*(4), 290-321.

Barbier, F., Henderson-Sellers, B., Parc-Lacayrelle, A. L., & Bruel, J.-M. (2003). Formalization of the Whole-Part Relationship in the Unified Modeling Language. *IEEE Transactions on Software Engineering, 29*(5), 459-470.

Beck, K. (2000). *Extreme Programming Explained - Embrace Change*. Boston et al.: Addison-Wesley.

Berner, S., Glinz, M., & Joos, S. (1999). A Classification of Stereotypes for Object-Oriented Modeling Languages. In R. France & B. Rumpe (Eds.), *UML '99 - The Unified Modeling Language - Beyond the Standard. Second International Conference, Fort Collins, CO, October 28-30, 1999* (Vol. 1723, pp. 249-264). Berlin et al.: Springer.

Bock, C. (2003). UML without Pictures. *IEEE Software, 20*(5), 33-35.

Booch, G. (1994). *Object-Oriented Analysis and Design with Applications* (2nd ed.). Redwood City, CA, USA: Benjamin/Cummings.

Booch, G. (1999). UML in Action. *Communications of the ACM, 42*(10), 26-29.

Booch, G. (2002). Growing the UML. *Software and Systems Modeling, 1*, 157-160.

Booch, G., Rumbaugh, J., & Jacobson, I. (1999). *The Unified Modeling Language User Guide*. Reading, MA, et al.: Addison-Wesley.

Bornberg-Bauer, E., & Paton, N. W. (2002). Conceptual data modelling for bioinformatics. *Briefings in Bioinformatics, 3*(2), 165-180.

Bruccoleri, M., Dieaga, S. N. L., & Perrone, G. (2003). An Object-Oriented Approach for Flexible Manufacturing Control Systems Analysis and Design Using the Unified Modeling Language. *The International Journal of Flexible Manufacturing Systems, 15*, 195-216.

Cengarle, M. V., & Knapp, A. (2004). OCL 1.4/5 vs. 2.0 Expressions - Formal semantics and expressiveness (*Online First Issue*). *Software and Systems Modeling*.

Champeaux, D. d. (2003). Extending and Shrinking UML. *Communications of the ACM, 46*(3), 11-12.

Chen, P. P.-S. (1976). The Entity-Relationship Model - Toward a Unified View of Data. *ACM Transactions on Database Systems, 1*(1), 9-36.

Coad, P., & Yourdon, E. (1991). *Object-Oriented Analysis*. Englewood Cliffs, N. J.: Prentice Hall.

D'Souza, D. F., & Wills, A. C. (1998). *Objects, Components, and Frameworks with UML - The Catalysis Approach*. Reading, MA, et al.: Addison-Wesley.

Dahanayake, A. (2003). Methodology Evaluation Framework for Component-Based System Development. *Journal of Database Management, 14*(1), 1-26.

Dobing, B., & Parsons, J. (2000). Understanding the Role of Use Cases in UML: A Review and Research Agenda. *Journal of Database Management, 11*(4), 28-36.

Dolk, D. R. (2000). Integrated model management in the data warehouse era. *European Journal of Operational Research, 122*, 199-218.

Dori, D. (2002). Why Significant UML Change is Unlikely. *Communications of the ACM, 45*(11), 82-85.

Douglass, B. P. (1998). *Real-Time UML: Developing Efficient Objects for Embedded Systems*. Reading, MA: Addison-Wesley.

Duddy, K. (2002). UML2 Must Enable a Family of Languages. *Communications of the ACM, 45*(11), 73-75.

Engels, G., Heckel, R., & Sauer, S. (2001). UML - A Universal Modeling Language? In M. Nielsen & D. Simpson (Eds.), *Application and Theory of Petri Nets 2000: 21st International Conference, ICATPN 2000 , June 2000, Aarhus, Denmark* (pp. 24-38). Berlin et al.: Springer.

Eriksson, H.-E., & Penker, M. (2000). *Business Modeling with UML - Business Patterns at Work*. New York et al.: John Wiley & Sons.

Evermann, J., & Wand, Y. (2001). Towards Ontologically Based Semantics for UML Constructs. In H. S. Kunii, S. Jajodia & A. Sølvberg (Eds.), *Conceptual Modeling - ER 2001 - 20th International Conference on Conceptual Modeling, Yokohama, Japan, November 27-30, 2001, Proceedings* (pp. 354-367). Berlin, Heidelberg: Springer.

Fettke, P., & Loos, P. (2003a). Classification of reference models - a methodology and its application. *Information Systems and e-Business Management, 1*(1), 35-53.

Fettke, P., & Loos, P. (2003b). Model Driven Architecture (MDA). *Wirtschaftsinformatik, 45*(5), 555-559.

Fettke, P., & Loos, P. (2003c). Specification of Business Components. In M. Aksit, M. Mezini & R. Unland (Eds.), *Objects, Components, Architectures, Services, and Applications for a Networked World - International Conference NetObjectDays, NODe 2002, Erfurt, Germany, October 7-10, 2002, Revised Papers* (Vol. 2591, pp. 62-75). Berlin et al.: Springer.

Firesmith, D., Henderson-Sellers, B., & Graham, I. (1998). *The OPEN Modeling Language (OML) reference manual*. Cambridge, UK, et al.: Cambridge University Press.

Fowler, M. (2004). *UML Distilled - A Brief Guide to the Standard Object Modeling Language* (3rd ed.). Boston et al.: Addison-Wesley.

Frank, U. (1997). *Towards a Standardization of Object-Oriented Modelling Languages?* (Working Paper No. 3). Koblenz, Germany: Institut für Wirtschaftsinformatik der Universität Koblenz Landau.

Frank, U. (1998). Object-Oriented Modelling Languages: State of the Art and Open Research Questions. In M. Schader & A. Korthaus (Eds.), *The Unified Modeling Language: Technical Aspects and Applications* (pp. 14-31). Heidelberg: Physica.

Frank, W., & Tyson, K. P. (2002). Be Clear, Clean, Concise. *Communications of the ACM, 45*(11), 79-81.

Frankel, D. S. (2003). *Model Driven Architecture - Applying MDA to Enterprise Computing*. Indianapolis, Indiana, USA: Wiley.

Fuentes, J. M., Quintana, V., Llorens, J., Génova, G., & Prieto-Díaz, R. (2003). Errors in the UML Metamodel? *ACM SIGSOFT Software Engineering Notes, 28*(6), 1-13.

Harel, D. (1987). Statecharts: A Visual Formalism for Complex Systems. *Science of Computer Programming, 8*, 231-274.

Henderson-Sellers, B. (2002). The Use of Subtypes and Stereotypes in the UML Model. *Journal of Database Management, 13*(2), 43-50.

Hofstede, A. H. M., & Weide, T. P. v. d. (1993). Formalisation of techniques: Chooping down the methodology jungle. *Information & Software Technology, 34*(1), 57-65.

Humphrey, W. S. (1995). *A Discipline for Software Engineering*. Reading, MA, et al.: Addison-Wesley.

Jackson, R. B., Liddle, S. W., & Woodfield, S. N. (1998). An Analysis of the Unified Modeling Language: UMl Compared with OSM. In S. Zamir (Ed.), *Handbook of Object Technology* (pp. 6-1 - 6-31). Boca Raton et al.: CRC Press.

Jacobson, I. (1987). *Object Oriented Development in an Industrial Environment.* Paper presented at the Conference on Object Oriented Programming Systems Languages and Applications (OOPSLA), Orlando, FL, USA.

Jacobson, I., Booch, G., & Rumbaugh, J. (1998). *The Unified Software Development Process*. Reading, MA, et al.: Addison-Wesley.

Jacobson, I., Christerson, M., Jonsson, P., & Övergaard, G. (1992). *Object-Oriented Software Engineering*. Reading, MA, et al.: Addison-Wesley.

Kobryn, C. (1999). UML 2001: A Standardization Odyssey. *Communications of the ACM, 42*(10), 29-37.

Kobryn, C. (2000). Modeling Components and Frameworks with UML. *Communications of the ACM, 43*(10), 31-38.

Kobryn, C. (2002). Will UML 2.0 Be Agile or Awkward? *Communications of the ACM, 45*(1), 107-110.

Kobryn, C. (2004). UML 3.0 and the future of modeling. *Software and Systems Modeling, 3*, 4-8.

Laitenberger, O., Atkinson, C., Schlich, M., & Emam, K. E. (2000). An experimental comparison of reading techniques for defect detection in UML design documents. *The Journal of Systems and Software, 53*(2000), 183-204.

Martin, J., & Odell, J. J. (1998). *Object-Oriented Methods: A Foundation - UML Edition* (2nd ed.). Upper Saddle River, NJ: Prentice Hall PTR.

McLeod, G., Halpin, T., Kangassalo, H., & Siau, K. (2001). *UML: A Critical Evaluation and Suggested Future.* Paper presented at the 34th Hawaii International Conference on System Sciences, Hawaii.

Medvidovic, N., Rosenblum, D. S., Redmiles, D. F., & Robbins, J. E. (2002). Modeling Software Architectures in the Unified Modeling Language. *ACM Transactions on Software Engineering and Methodology, 11*(1), 2-57.

Mellor, S. J. (2002). Make Models be Assets. *Communications of the ACM, 45*(11), 76-78.

Mellor, S. J., & Balcer, M. J. (2002). *Executable UML: A Foundation for Model-Driven Architecture*. Boston et al.: Addison Wesley.

Meyer, B. (1997). *Object-Oriented Software Construction* (2nd ed.). Upper Saddle River, NJ, USA: Prentice Hall.

Miller, J. (2002). What UML Should Be. *Communications of the ACM, 45*(11), 67-69.

Mylopoulos, J. (1998). Information Modeling in the Time of the Revolution. *Information Systems, 23*(3/4), 127-155.

OMG. (1999). *Analysis and Design Platform Task Force, White Paper on Profile mechanisms, Version 1.0, ad/99-04-07*. Framingham, MA, USA.

OMG. (2002). *Meta Object Facility (MOF) Specification, Version 1.4*. Needham, MA, USA.

OMG. (2003a). *Common Warehouse Metamodel (CWM) Specification, Version 1.1, formal/03-03-02*. Needham, MA, USA.

OMG. (2003b). *MDA Guide Version 1.0* (No. omg/2003-05-01). o. O.

OMG. (2003c). *OCL 2.0, Final Adopted Specification, ptc/03-10-14*. Needham.

OMG. (2003d). *OMG Unified Modeling Language Specification: Version 1.5, formal/03-03-01*. Needham, MA, USA.

OMG. (2003e). *UML 2.0 Standard Officially Adopted at OMG Technical Meeting in Paris*. Retrieved 2004-03-01, from http://www.omg.org/news/releases/pr2003/6-12-032.htm

OMG. (2003f). *Unified Modeling Language: Infrastructure, Version 2.0, Final Adopted Specification, ptc/03-09-15*. Needham.

OMG. (2003g). *Unified Modeling Language: Superstructure, Version 2.0, Final Adopted Specification, ptc/03-08-02*. Needham.

OMG. (2003h). *XML Metadata Interchange (XMI) Specification, Version 2.0, formal/03-05-02*. Needham, MA, USA.

OMG. (2004a). *Business Enterprise Integration Domain Taks Force*. Retrieved 2004-03-01, 2004, from http://bei.omg.org/

OMG. (2004b). *Finance Domain Taks Force*. Retrieved 2004-03-01, from http://bei.omg.org/

OMG. (2004c). *UML Tools*. Retrieved 2004-03-01, 2004, from http://www.omg.org/uml

Opdahl, A. L., & Henderson-Sellers, B. (2002). Ontological Evaluation of the UML Using the Bunge-Wand-Weber Model. *Software and Systems Modeling, 1*(1), 43-67.

Robbins, J. E., & Redmiles, D. F. (2000). Cognitive support, UML adherence, and XMI interchange in Argo/UML. *Information and Software Technology, 42*, 79-89.

Rumbaugh, J., Blaha, M., Premerlani, W., Eddy, F., & Lorensen, W. (1991). *Object-Oriented Modeling and Design*. Englewood Cliffs, N. J.: Prentice Hall.

Rumbaugh, J., Jacobson, I., & Booch, G. (1998). *The Unified Modeling Language Reference Manual*: Addison-Wesley.

Saleh, K. (2002). Documenting electronic commerce systems and software using the unified modeling language. *Information and Software Technology, 44*(2002), 303-311.

Schader, M., & Korthaus, A. (Eds.). (1998). *The Unified Modeling Language: Technical Aspects and Applications*. Heidelberg: Physica.

Schneider, G., & Winters, J. (2001). *Applying Use Cases - A Practical Guide* (2nd ed.). Boston et al.: Addison-Wesley.

Selic, B., Ramackers, G., & Kobryn, C. (2002). Evolution, not Revolution. *Communications of the ACM, 45*(11), 70-72.

Selonen, P., Koskimies, K., & Sakkinen, M. (2003). Transformations between UML diagrams. *Journal of Database Management, 14*(3), 37-55.

Shen, Z., & Siau, K. (2003). *An Empirical Evaluation of UML Notational Elements Using a Concept Mapping Approach.* Paper presented at the Twenty-Fourth International Conference on Information Systems, Seattle, Washington, USA.

Shlaer, S., & Mellor, S. J. (1988). *Object-Oriented Systems Analysis: Modeling the World in Data*. Englewood Cliffs, NJ: Yourdon.

Siau, K., & Cao, Q. (2001). Unified Modeling Language (UML) - A Complexity Analysis. *Journal of Database Management, 12*(1), 26-34.

Siau, K., & Halpin, T. (Eds.). (2001). *Unified Modeling Language: Systems Analysis, Design and Development Issues*. Hershey et al.: Idea Group.

Thomas, D. (2002). UML - Unified or Universal Modeling Language? *Journal of Object Technology, 2*(1), 7-12.

Wand, Y., & Weber, R. (2002). Research Commentary: Information Systems and Conceptual Modeling - A Research Agenda. *Information Systems Research, 13*(4), 363-377.

Wang, S. (2001). *Experiences with the Unified Modeling Language (UML).* Paper presented at the Seventh Americas Conference on Information Systems (AMCIS) 2001.

Warmer, J. v., & Kleppe, A. (2003). *The Object Constraint Language - Getting your models ready for MDA* (2nd ed.): Addison-Wesley.

Wirfs-Brock, R., Wilkerson, B., & Wiener, L. (1990). *Designing object-oriented Software*. Englewood Cliffs, NJ: Prentice Hall.