UMLsec: Extending UML for Secure Systems Development*

Jan Jürjens**

Software & Systems Engineering, Dep. of Informatics
Munich University of Technology, Germany

Abstract. Developing secure-critical systems is difficult and there are many well-known examples of security weaknesses exploited in practice. Thus a sound methodology supporting secure systems development is urgently needed.
Our aim is to aid the difficult task of developing security-critical systems in an approach based on the notation of the Unified Modeling Language. We present the extension UMLsec of UML that allows to express security-relevant information within the diagrams in a system specification. UMLsec is defined in form of a UML profile using the standard UML extension mechanisms. In particular, the associated constraints give criteria to evaluate the security aspects of a system design, by referring to a formal semantics of a simplified fragment of UML. We demonstrate the concepts with examples.

1 Introduction

Modern society and modern economies rely on infrastructures for communication, finance, energy distribution, and transportation. These infrastructures depend increasingly on networked information systems. Attacks against these systems can threaten the economical or even physical well-being of people and organizations. Due to the widespread interconnection of information systems, attacks can be waged anonymously and from a safe distance. Many security incidents have been reported, sometimes with potentially quite severe consequences.

Many problems with security-critical systems arise from the fact that their developers do not always have a strong background in computer security. This is problematic since in practice, security is compromised most often not by breaking mechanisms such as encryption or security protocols, but by exploiting weaknesses in the way they are being used. Security mechanisms cannot be "blindly" inserted into a security-critical

* Supported by the German Ministry of Economics within the FairPay project.
** http://www.jurjens.de/jan – juerjens@in.tum.de

system, but the overall system development must take security aspects into account.

Furthermore, sometimes security mechanisms (such as security protocols) cannot be used off-the-shelf, but have to be designed specifically to satisfy given requirements (for example on the hardware). Such mechanisms are notoriously hard to design correctly, even for experts, as many examples of protocols designed by experts that were later found to contains flaws show.

Any support to aid secure systems development is thus dearly needed. In particular, it would be desirable to consider security aspects already in the design phase, before a system is actually implemented, since removing security flaws in the design phase saves cost and time.

This has motivated a significant amount of research into using formal methods for secure systems development. However, part of the difficulty of security-critical systems development is that correctness is often in conflict to cost. Where thorough methods of system design pose high cost through personnel training and use, they are all too often avoided.

The Unified Modeling Language (UML, [RJB99, UML01], the de facto industry-standard in object-oriented modeling) offers an unprecedented opportunity for high-quality critical systems development that is feasible in an industrial context.

- As the de facto standard in industrial modeling, a large number of developers is trained in UML.
- Compared to previous notations with a user community of comparable size, UML is relatively precisely defined.

To support using UML for secure systems development, we give an extension, UMLsec, of the UML, following a suggestion in [DS00]. This way we encapsulate knowledge on prudent security engineering and thereby make it available to developers which may not be specialized in security. One can also go further by checking whether the constraints associated with the UMLsec stereotypes are fulfilled in a given specification, if desired by performing a formal analysis.

After presenting some background on distributed system security and on UML extension mechanisms in the following subsections, we explain how to formally evaluate UML specifications for security requirements in Section 2. We introduce UMLsec in Section 3. We give an account of experiences in using UMLsec in Section 3. After pointing to related work, we indicate future work and end with a conclusion. Due to space restrictions, we present only a representative fragment of UMLsec. A complete account can be found in [Jür02c].

Distributed System Security We exemplarily explain a few important recurring security requirements of distributed object-oriented systems which are encapsulated in UML stereotypes and tags in the UMLsec profile by associating formalizations of these requirements (referring to the formal semantics) as constraints with the stereotypes. The formalizations are obtained following standard approaches to formal security analysis. There are other general security requirements (such as availability and integrity) also covered by the UMLsec approach which have to be omitted. More details can be found in [Jür02c].

Fair exchange When trading goods electronically, the requirement *fair exchange* postulates that the trade is performed in a way that prevents both parties from cheating. In our context here we mean more specifically the requirement that after a prepayment the buyer either receives the purchased good or is able to reclaim the money.

Secrecy/confidentiality One of the main data security requirements is secrecy (or confidentiality), meaning that some information will become known only to legitimate parties.

Secure information flow Sometimes even a *partial* leakage of information must be prevented (for example, leaking one half of a cryptographic key may give enough information to recover the other half by brute force). The notion of *secure information flow* ensures that where trusted parts of a system interact with untrusted parts, there is not even a *partial* leakage of secret information from the trusted to the untrusted part.

Secure communication link This requirement ensures that the physical communication links between different parts of the system give the required security guarantee regarding a given adversary model. For example, a local area network (LAN) is a secure link with respect to secrecy against *outsider* attackers.

UML extension mechanisms The three main "lightweight" extension mechanisms are stereotypes, tagged values and constraints. Stereotypes, in double angle brackets, define new types of modeling elements extending the semantics of existing types in the UML metamodel. A tagged value is a name-value pair in curly brackets associating data with model elements. Constraints may also be attached. A UML extension collects stereotypes, tagged values, and constraints into a *profile*. For UMLsec, we give validation rules evaluating a model against included security requirements.

For this we extend a formal semantics for the used fragment of UML in a modular way with a formal notion of an adversary.

Related Work To our knowledge, this is the first work proposing an extension of UML for the development of security-critical systems. [Jür01] gave some initial ideas on how to use UML to develop security-critical systems (without actually defining an extension profile). For more material on UMLsec see http://www4.in.tum.de/~umlsec. [Jür02d] gives a process for applying UMLsec using goal-trees profiting from work in [MCY99].

Distributed system modeling with UML is considered in [Kob01]. [FH97] defines role-based access control rights from object-oriented use cases. [WW01] gives an approach similar to ours but using the notation of the CASE tool AutoFocus. Also relevant is the work towards a formal semantics of UML including [GPP98, SW98, KER99, BLMF00, Whi00, EHHS00, MC01, Ste01].

2 Security evaluation of UML diagrams

We briefly give an idea how the constraints used in the UMLsec profile can be checked in a precise and well-defined way. More details can be found in [Jür02c].

Outline of formal semantics For some of the constraints used to define the UMLsec extensions we need to refer to a precisely defined semantics of behavioral aspects. For security analysis, the security-relevant information from the security-oriented stereotypes is then incorporated (cf. Section 2).

Our formal semantics of a simplified fragment of UML builds on [BCR00]; parts of it are in [Jür02a, Jür02b]. It includes activity diagrams, statecharts, sequence diagrams, static structure diagrams, deployment diagrams, and subsystems, simplified to keep a formal treatment that is necessary for some of the more subtle security requirements feasible. The subsystems integrate the information between the different kinds of diagrams and between different parts of the system specification. We only outline the basic concepts, a complete account is in [Jür02c].

In UML the objects or components communicate through messages received in their input queues and released to their output queues. Thus for each component C of a given system, our semantics defines a function $[\![C]\!]()$ which

- takes a multi-set I of input messages and a component state S and
- outputs a set $[\![C]\!](I, S)$ of pairs (O, T) where O is a multi-set of output messages and T the new component state (it is a *set* of pairs because of the non-determinism that may arise)

together with an *initial state* S_0 of the component.

The behavioral semantics $[\![D]\!]()$ of a statechart diagram D models the run-to-completion semantics of UML statecharts. As a special case, this gives us the semantics for activity diagrams. Given a sequence diagram $\mathcal{S}$, we define the behavior $[\![\mathcal{S}.C]\!]()$ of each contained component C.

Subsystems group together diagrams describing different parts of a system: a system component $\mathcal{C}$ given by a subsystem $\mathcal{S}$ may contain subcomponents $\mathcal{C}_1, \ldots, \mathcal{C}_n$. The behavioral interpretation $[\![\mathcal{S}]\!]()$ of $\mathcal{S}$ is defined as follows:

(1) It takes a multi-set of input events.
(2) The events are distributed from the input multi-set and the link queues connecting the subcomponents and given as arguments to the functions defining the behavior of the intended recipients in $\mathcal{S}$.
(3) The output messages from these functions are distributed to the link queues of the links connecting the sender of a message to the receiver, or given as the output from $[\![\mathcal{S}]\!]()$ when the receiver is not part of $\mathcal{S}$.

When performing security analysis, after the last step, the adversary model may modify the contents of the link queues in a certain way explained in Section 2.

Security Analysis For a security analysis of a given UMLsec subsystem specification $\mathcal{S}$, we need to model potential adversary behavior. We model specific types of adversaries that can attack different parts of the system in a specified way. For this we assume a function $\mathsf{Threats}_A(s)$ which takes an *adversary type* A and a stereotype s and returns a subset of $\{\mathsf{delete}, \mathsf{read}, \mathsf{insert}\}$. Then we model the actual behavior of an adversary of type A as a *type A adversary function* that non-deterministically maps the contents of the link queues in $\mathcal{S}$ and a state S to the new contents of the link queues in $\mathcal{S}$ and a new state T:

- the contents of links stereotyped s where $\mathsf{delete} \in \mathsf{Threats}_A(s)$ may be mapped to $\emptyset$ and
- the contents of links stereotyped s where $\mathsf{insert} \in \mathsf{Threats}_A(s)$ may be enlarged by elements from the contents of links stereotyped t where $\mathsf{read} \in \mathsf{Threats}_A(t)$.

The adversary types define which actions an adversary may apply to a communication link with a given stereotype. delete means that the adversary may delete the messages in the corresponding link queue, read allows him to read the messages in the link queue, and insert allows him to insert messages in the link queue.

To evaluate the security of the system with respect to the given type of adversary, we define the *execution of the subsystem $\mathcal{S}$ in presence of an adversary of type A* to be the function $[\![\mathcal{S}]\!]_A()$ defined from $[\![\mathcal{S}]\!]()$ by applying the adversary function to the link queues as a fourth step in the definition of $[\![\mathcal{S}]\!]()$ as follows:

(4) The type A adversary function is applied to the link queues as detailed above.

The UMLsec profile makes use of a formalization of the security requirement *secrecy* following one of the standard approaches in formal methods: It relies on the idea that a specification preserves the secrecy of some data d if the system never sends out any information from which d could be derived, even in interaction with an adversary (where the *knowledge set* collects the information gained by an adversary).

Definition 1. *We say that a subsystem $\mathcal{S}$* preserves the secrecy *of an expression E from adversaries of type A if E never appears in the knowledge set of A during execution of $[\![\mathcal{S}]\!]_A()$.*

A secrecy like requirement that gives protection also against partial flow of information is that of *down-flow prevention*. Given a set of messages H and a sequence $\boldsymbol{m}$ of event multi-sets, we write $\boldsymbol{m}\lfloor_H$ for the sequence of event multi-sets derived from those in $\boldsymbol{m}$ by deleting all events the message names of which are in H. For a set M of sequences of messages, we define $M\lfloor_H \stackrel{\text{def}}{=} \{\boldsymbol{m}\lfloor_H : \boldsymbol{m} \in M\}$.

Definition 2. *Given a subsystem $\mathcal{S}$ and a set of messages H, we say that $\mathcal{S}$* prevents down-flow *with respect to H if for any two sequences $\mathbf{i}, \mathbf{j}$ of input event multi-sets, $\mathbf{i}_H = \mathbf{j}\lfloor_H$ implies $[\![\mathcal{S}]\!]_A(\boldsymbol{i})\lfloor_H = [\![\mathcal{S}]\!]_A(\boldsymbol{j})\lfloor_H$.*

Intuitively, preventing down-flow means that an output not assumed to be secret should in no way depend on secret inputs.

3 The UMLsec extension

We can only describe a small fragment of the UMLsec profile to illustrate the idea, in order to keep the presentation concise. The fragment was

chosen to be representative for the other UMLsec concepts that had to be omitted (for example, *integrity* can be treated analogously to *secrecy*). A complete account can be found in [Jür02c]. We give the profile following the structure in [UML01].

Applicable subset The profile concerns all of UML.

Stereotypes, tagged values and constraints In Figure 1 we give some of the stereotypes from UMLsec, together with their tags and constraints, following the notation used in [UML01, 3-59]. The stereotypes do not have parents. The constraints, which in the table are only named briefly, are formulated (in plain mathematical language) and explained in the remainder of the section. Figure 2 gives the corresponding tags (which are all DataTags). Note that some of the stereotypes on subsystems refer to stereotypes on model elements contained in the subsystems. For example, the constraint of the «data security» stereotype refers to contained objects stereotyped «critical» (which in turn have tags {secret}). The relations between the elements of the tables are explained below in detail.

Stereotype	Base Class	Tags	Constraints	Description
Internet	link			Internet connection
encrypted	link			encrypted connection
LAN	link			LAN connection
secure links	subsystem		dependency security matched by links	enforces secure communication links
secrecy	dependency			assumes secrecy
secure dependency	subsystem		«call», «send» respect data security	structural interaction data security
critical	object	secret		critical object
no down-flow	subsystem		prevents down-flow	information flow
data security	subsystem		provides secrecy	basic datasec requirements
fair exchange	package	start,stop	after start eventually reach stop	enforce fair exchange

Fig. 1. UMLsec stereotypes (excerpt)

Prerequisite profiles UMLsec requires no prerequisite Profiles.

Well-formedness rules We explain the stereotypes and tags given in Figures 1 and 2 and give examples. The constraints are parameterized over

Tag	Stereotype	Type	Multipl.	Description
secret	critical	String	*	secret data
start	fair exchange	$\mathcal{P}$(String)	1	start states
stop	fair exchange	$\mathcal{P}$(String)	1	stop states

Stereotype	$\mathsf{Threats}_{default}()$
Internet	{delete,read,insert}
encrypted	{delete}
LAN	$\emptyset$

Fig. 2. UMLsec tags (excerpt); Threats from the *default* attacker

the adversary type with respect to which the security requirements should hold; we thus fix an adversary type A to be used in the following. By their nature, some of the constraints can be enforced at the level of abstract syntax (such as «secure links»), while others refer to the formal definitions in Section 2 (such as «no down − flow»). Note that even checking the latter can be mechanized given appropriate tool-support (for example along the lines of [HJGP99]).

Internet, encrypted, LAN These stereotypes on links in deployment diagrams denote the respective kinds of communication links. We require that each link carries at most one of these stereotypes. For each adversary type A, we have a function $\mathsf{Threats}_A(s)$ from each stereotype $s \in \{$«encrypted», «LAN», «Internet»$\}$ to a set of strings $\mathsf{Threats}_A(s) \subseteq \{$delete, read, insert$\}$. This way we can evaluate UML specifications using the approach explained in Section 2. We make use of this for the constraints of the remaining stereotypes of the profile.

As an example for a threat function, Figure 2 gives the one for the *default* type of attacker, which represents an outsider adversary with modest capability.

secure links This stereotype, which may label subsystems, is used to ensure that security requirements on the communication are met by the physical layer. More precisely, the constraint enforces that for each dependency d stereotyped «secrecy» between subsystems or objects on different nodes n, m, we have a communication link l between n and m with stereotype s such that $\mathsf{read} \notin \mathsf{Threats}_A(s)$ (again a more complete definition also involving integrity can be found in [Jür02c]).

Example In Figure 3, given the *default* adversary type, the constraint for the stereotype «secure links» is violated: The model does not provide communication secrecy against the *default* adversary, because the Internet communication link between web-server and client does not provide the needed security level according to the $\mathsf{Threats}_{default}(Internet)$ scenario.

secrecy «call» or «send» dependencies in object or component diagrams stereotyped «secrecy» are supposed to provide secrecy for the data that

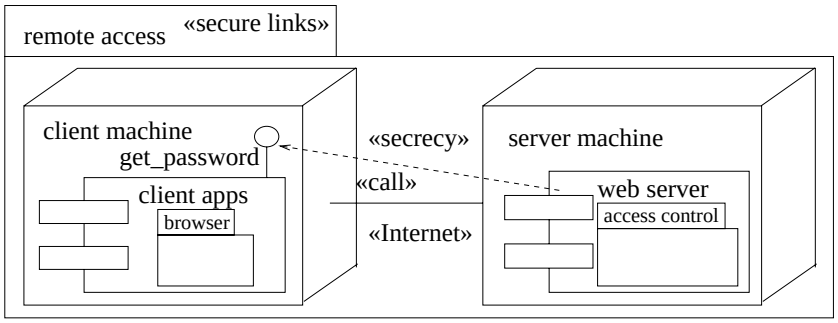


Fig. 3. Example *secure links* usage

is sent along them as arguments or return values of operations or signals. This stereotype is used in the constraint for the stereotype « secure links ».

secure dependency This stereotype, used to label subsystems containing object diagrams or static structure diagrams, ensures that the « call » and « send » dependencies between objects or subsystems respect the security requirements on the data that may be communicated along them. More exactly, the constraint enforced by this stereotype is that if there is a « call » or « send » dependency from an object (or subsystem) C to an object (or subsystem) D then the following conditions are fulfilled.

- For any message name n offered by D, n appears in the tag {secret} in C if and only if it does so in D.
- If a message name offered by D appears in the tag {secret} in C then the dependency is stereotyped « secrecy ».

Example Figure 4 shows a key generation subsystem stereotyped with the requirement « secure dependency ». The given specification violates the constraint for this stereotype, since Random generator and the « call » dependency do not provide the security levels for random() required by Key generator.

critical This stereotype labels objects whose instances are critical in some way, as specified by the associated tag {secret}, the values of which are data values or attributes of the current object the secrecy of which are supposed to be protected. This protection is enforced by the constraints of the stereotypes « data security » and « no down − flow » (depending on the degree of secrecy required) which label subsystems that contain « critical » objects.

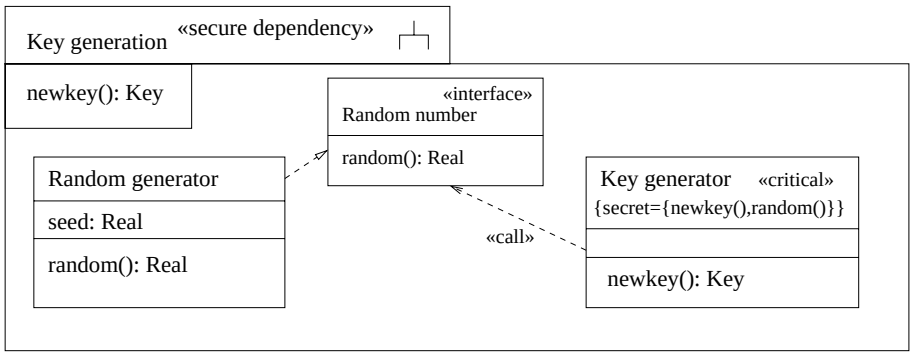


Fig. 4. Key generation subsystem

no down-flow This stereotype of subsystems enforces secure information flow by making use of the associated tag {secret}. More precisely, the «no down − flow» constraint is that the stereotyped subsystem prevents down-flow with respect to the messages and their return messages specified in {secret}, as defined in Definition 2.

Example The example in Figure 5 shows a bank account data object that allows its secret balance to be read using the operation rb() whose return value is also secret, and written using wb(x). If the balance is over 10000, the object is in a state ExtraService, otherwise in NoExtraService. The state of the object can be queried using the operation rx(). The data object is supposed to be prevented from indirectly leaking out any partial information about secret data via non-secret data, as specified by the stereotype «no down − flow». The given specification violates this requirement, since partial information about the input of the secret operation wb() is leaked out via the return value of the non-secret operation rx().

data security This stereotype labeling subsystems has the following constraint. The subsystem behavior respects the data security requirements given by the stereotype «critical» and the associated tags, with respect to the threat scenario arising from the deployment diagram. More precisely, the constraint is that the stereotyped subsystem preserves the secrecy of the data designated by the tag {secret} against adversaries of type A as defined in Definition 1.

Example The example in Figure 6 shows the specification of a simple security protocol. The sender requests the public key K together with the certificate $\mathcal{S}ign_{K_{CA}}(rcv :: K)$ certifying authenticity of the key from the

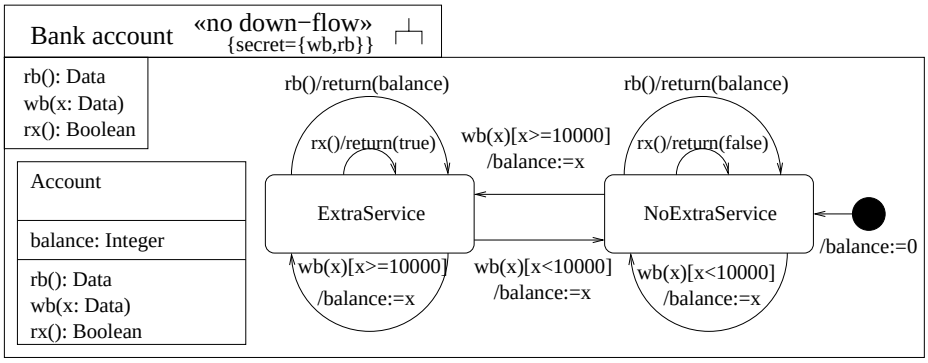


Fig. 5. Bank account data object

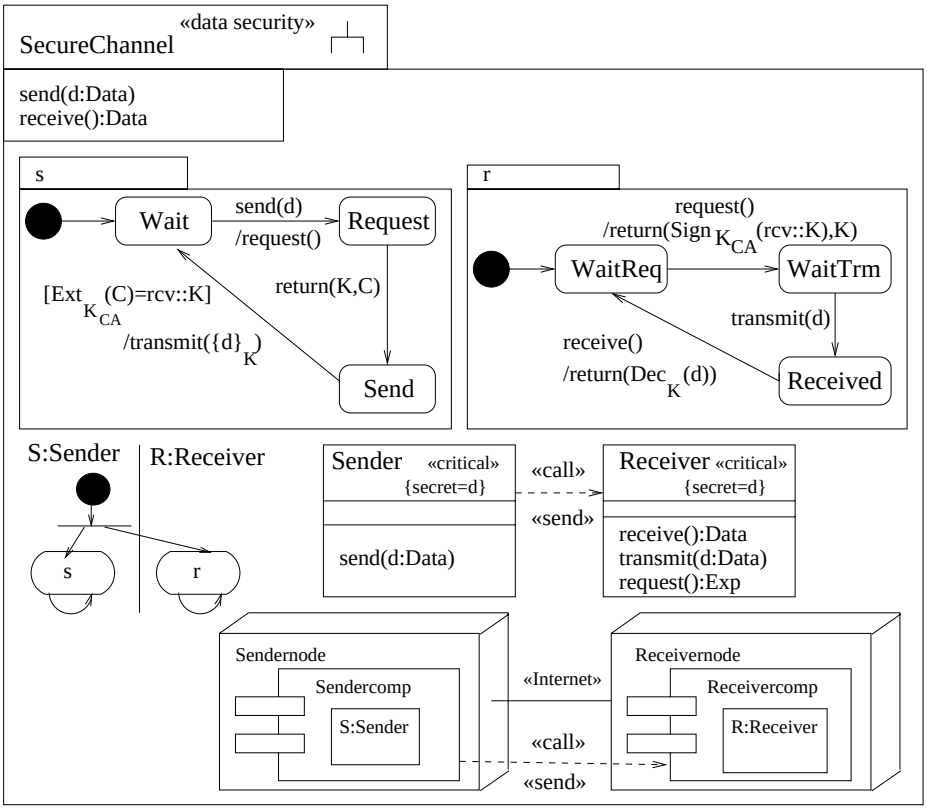


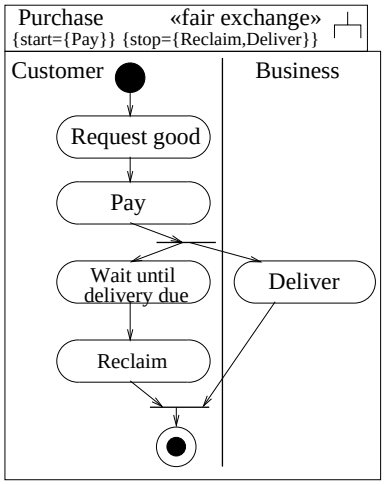
Fig. 6. Security protocol

receiver and sends the data d back encrypted under K (here $\{M\}_K$ is the encryption of the message M with the key K, $\mathcal{D}ec_K(C)$ is the decryption of the ciphertext C using K, $\mathcal{S}ign_K(M)$ is the signature of the message M with K, and $\mathcal{E}xt_K(S)$ is the extraction of the data from the signature using K). Assuming the *default* adversary type and by referring to the adversary model outlined in Section 2, one can establish that the secrecy of d is preserved.

fair exchange This stereotype of subsystems has associated tags {start} and {stop} taking sets of names of states as values ($\mathcal{P}(X)$ denotes the set of subsets of a set X). The associated constraint requires that, whenever a {start} state in the contained activity diagram is reached, then eventually a {stop} state will be reached. This allows one to formalize the specific fair exchange requirement explained in Section 1. This is formalized for a given subsystem $\mathcal{S}$ as follows. $\mathcal{S}$ fulfills the constraint of «fair exchange» if for every adversary *adv* of type A and every sequence of input event multi-set $I_1, \ldots, I_n$, the following implication holds: If the function associated with $\mathcal{S}$ reaches a state specified by {start}, then subsequently it eventually reaches a state specified by {stop}.

Note that this requirement cannot be ensured for systems which an attacker can stop completely.

Example The figure gives a subsystem describing the following situation: a customer buys a good from a business. The semantics of the stereotype «fair exchange» is, intuitively, that the actions listed in the tags {start} and {stop} should be linked in the sense that if one of the former kind is executed then eventually one of the latter kind will be.
This would entail that, once the customer has paid, he is either delivered the order by the due date, or is able to reclaim the payment on that date.



Requirements on an UML extension for development of security-critical systems We formulate what we consider necessary properties of an UML extension for secure systems development and discuss whether they are fulfilled by UMLsec. Following the format of the OMG Requests

for Proposals (RFPs) we distinguish mandatory and optional requirements.

Mandatory requirements

Security requirements Formalizations of basic security requirements are provided via stereotypes, such as «secrecy».

Threat scenarios Threat scenarios are incorporated using the formal semantics and depending on the modeled underlying physical layer via the sets $\mathsf{Threats}_{adv}(ster)$ of actions available to the adversary of kind *adv*.

Security concepts We have shown how to incorporate security concepts such as encrypted communication links (using threat scenarios, in this case).

Security mechanisms In further work we demonstrate how to incorporate security mechanisms at the example of Java security architecture mechanisms such as guarded objects (see [Jür02c]).

Security primitives Security primitives such as encryption are built in.

Underlying physical security Physical security can be addressed as demonstrated by the stereotype «secure link» in deployment diagrams.

Optional requirements It would be very useful to include domain-specific security knowledge. — UMLsec has been used in the application domain of smart-card based systems and of Java security and CORBA security (for related material see http://www4.in.tum.de/~umlsec).

Experience The method proposed here has been successfully applied in security consulting, for example in an evaluation of the Common Electronic Purse Specifications under development by Visa International and others, in the project FairPay funded by the German Ministry of Economics, and in projects with a large German bank. In particular, these experiences show that the used adversary model is adequate for use in practice.

4 Conclusion and Future Work

We proposed an extension of UML, called UMLsec, to aid development of security-critical systems. Given the current state of computer security in practice, with many vulnerabilities reported continually, this seems to be a useful line of research, since it enables developers with background in security to make use of security engineering knowledge encapsulated

in a widely used design notation. Since the behavioral parts of UMLsec are considered with a formal semantics, this allows a formal evaluation (parts of which may be mechanized). Thus even security experts undertaking a formal evaluation for certification purposes may profit from the possibility of using a specification language that may be more usable than some traditional formal methods. Since UML specifications may already exist independently from the formal evaluation, this should reduce cost of certification. Note that one may use UMLsec without having to refer to a formal semantics for UML. In that case, the constraints for the security requirements would have to be checked by a CASE tool and explained to the user informally. It is however beneficial to have a formal reference that tool providers can refer to if necessary; this is why [Jür02c] provides a formal semantics for the used fragment of UML.

For this line of research to be of practical value it is important to develop tool support, which is under way (by using XMI to integrate a UML editor with the CASE tool AutoFocus). This tool (with a UML-like notation) has already proven to be useful in several security-related industry projects (cf. e.g. [JW01]); an extension to the actual UML notation would have the additional benefit of a standardised notation.

Some more guidance on how to employ UMLsec in the development context is given in [Jür02c]. Currently it is investigated how to use use cases in the context of UMLsec for security requirements capture.

Our approach could also be employed profitably for the development and analysis of critical systems with respect to other non-functional requirements, in which case the current adversary model has to be modified to simulate other environment influences (for example, random insertion of delays into communication links in the case of quality-of-service requirements).

Acknowledgements This line of research applying UML to security has benefitted from comments especially from P. Stevens, G. Lowe, and B. Rumpe. Comments from R. Sandner, G. Wimmel, G. Popp, and P. Shabalin on a draft of this paper are very gratefully acknowledged.

References

[ACM02] ACM. *Symposium of Applied Computing 2002*, Madrid, March 11–14 2002.

[BCR00] E. Börger, A. Cavarra, and E. Riccobene. Modeling the dynamics of UML State Machines. In *ASMs*, volume 1912 of *LNCS*. Springer, 2000.

[BLMF00] J.-Michel Bruel, J. Lilius, A. Moreira, and R.B. France. Defining Precise Semantics for UML. In *ECOOP'2000 Workshop Reader*, volume 1964 of *LNCS*. Springer, 2000.

[DS00] P. Devanbu and S. Stubblebine. Software engineering for security: a roadmap. In *The Future of Software Engineering (ICSE 2000)*, pages 227–239, 2000.

[EHHS00] G. Engels, J. Hausmann, R. Heckel, and S. Sauer. Dynamic meta-modeling. In Evans et al. [EKS00], pages 323–337.

[EKS00] A. Evans, S. Kent, and B. Selic, editors. *The Unified Modeling Language: Advancing the Standard (UML'2000)*, volume 1939 of *LNCS*. Springer, 2000.

[FH97] E.B. Fernandez and J.C. Hawkins. Determining role rights from use cases. In *Workshop on Role-Based Access Control*, pages 121–125. ACM, 1997.

[GPP98] M. Gogolla and F. Parisi-Presicce. State diagrams in UML. In *PSMT'98*. TU München, TUM-I9803, 1998.

[HJGP99] W.-M. Ho, J.-M. Jézéquel, A. Le Guennec, and F. Pennaneac'h. UMLAUT: an extendible UML transformation framework. In *ASE*, 1999.

[Huß01] H. Hußmann, editor. *Fundamental Approaches to Software Engineering (FASE, 4th International Conference)*, volume 2029 of *LNCS*. Springer, 2001.

[Jür01] J. Jürjens. Towards development of secure systems using UML. In Hußmann [Huß01], pages 187–200.

[Jür02a] J. Jürjens. A UML statecharts semantics with message-passing. In *Symposium of Applied Computing 2002* [ACM02], pages 1009–1013.

[Jür02b] J. Jürjens. Formal Semantics for Interacting UML subsystems. In *FMOODS 2002*, pages 29–44. IFIP, Kluwer, 2002.

[Jür02c] J. Jürjens. *Principles for Secure Systems Design*. PhD thesis, Oxford University Computing Laboratory, Trinity Term 2002. Submitted.

[Jür02d] J. Jürjens. Using UMLsec and Goal-Trees for Secure Systems Development. In *Symposium of Applied Computing 2002* [ACM02], pages 1026–1031.

[JW01] J. Jürjens and G. Wimmel. Security modelling for electronic commerce: The Common Electronic Purse Specifications. In *I3E 2001*, pages 489–506. IFIP, Kluwer, 2001.

[KER99] S. Kent, A. Evans, and B. Rumpe. UML Semantics FAQ. In *ECOOP'99 Workshop Reader*, volume 1743 of *LNCS*. Springer, 1999.

[Kob01] C. Kobryn. Modeling Distributed Applications with UML, Part IV. In J. Siegel, editor, *Quick CORBA 3*, chapter 1. Wiley, 2001.

[MC01] W.E. McUmber and B.H.C. Cheng. A Generic Framework for Formalizing UML. In *ICSE*. IEEE Computer Society, 2001.

[MCY99] J. Mylopoulos, L. Chung, and E. Yu. From object-oriented to goal-oriented requirements analysis. *Communications of the ACM*, 42(1):31–37, 1999.

[RJB99] J. Rumbaugh, I. Jacobson, and G. Booch. *The Unified Modeling Language Reference Manual*. Addison-Wesley, 1999.

[Ste01] P. Stevens. On use cases and their relationships in the Unified Modelling Language. In Hußmann [Huß01], pages 140–155.

[SW98] A. Schürr and A. Winter. Formal Definition of UML's Package Concept. In *UML – Technical Aspects and Applications*, pages 144–159, 1998.

[UML01] UML Revision Task Force. OMG UML Specification v. 1.4. OMG Document ad/01-09-67. Available at http://www.omg.org/uml, 2001.

[Whi00] J. Whittle. Formal approaches to systems analysis using UML: An overview. *Journal of Database Management*, 11(4):4–13, 2000.

[WW01] G. Wimmel and A. Wißpeitner. Extended description techniques for security engineering. In *IFIP SEC 2001*. Kluwer, 2001.