Formal Development and Verification of Security-Critical Systems with UML (Position Paper)

Jan Jürjens*

Computing Laboratory, University of Oxford, GB

Abstract. We sketch ongoing work on using a formal fragment of the object-oriented modelling language UML (Unified Modeling Language) in development and verification of secure systems.

We sketch some current challenges for the formal (in particular automated) development and verification of (in particular) security-critical systems and note how some of our work tries to address these.

When trying to use formal (and in particular automated) techniques to develop or verify real-life security-critical systems whose specifications can easily consist of several hundred pages or more (e.g. the Common Electronic Purse Specifications (CEPS) considered in [Jü01c]) one faces a number of problems in scaling up existing approaches:

1. Usually a formal specification of the system is not available. Constructing it requires expert knowledge and can be very time-consuming. Currently a large part of effort both in verifying and in implementing specifications is wasted since these are often formulated imprecisely and unintelligibly [Pau98].

2. It is usually only feasible to construct the specification of a small security-critical part of the system (e.g. security protocols [Low96, LR97, RSG+01, SCW00], where formal analysis has in fact been quite successful). The boundaries of these components with the rest of the system need to be carefully examined, e.g. wrt implicit assumptions on the system context or the underlying physical layer [Gol00, Aba00].

In practice, most attacks do not address vulnerabilities in dedicated mechanisms (such as security protocols), but in the way these are employed in the system context [And01]. An example is given in [JW01a], where a vulnerability arises from the intended use of the CEPS payment scheme over the Internet.

* http://www.jurjens.de/jan  -  jan@comlab.ox.ac.uk  -  Supported by the Studienstiftung des deutschen Volkes and the Computing Laboratory.
(3) Stepwise development and modular reasoning are hindered by the fact that many security properties are not preserved under refinement and not compositional.

(4) Even when specifying small components, one often has to give a simplifying account. In particular, in symbolic modelling cryptographic operations one abstracts from the possibility that an adversary may simply guess secret values.

(5) Often, vulnerabilities arise from flaws in the implementation rather than the design.

We try to address some of these issues in previous, current and future work as follows:

(1) As a specification language we propose to use a formal fragment of UML [OMG99], extended with security-relevant primitives using the UML extension mechanisms [Jür01c, Jür01f, Jür01b]. As UML is de-facto industry-standard in object-oriented modelling, in some cases a specification of the system in it may already be available, or at least there may be knowledge available on the side of the developers how to construct the specification (additionally, there is work on deriving Java code from UML specifications [EHSW99] and vs. [KG01], narrowing the gap between specification- and software-based verification). For mechanical verification one may take advantage of work giving UML diagrams a formal semantics\(^1\) in CSP, allowing to use FDR2 [BD00, Cri01], and of other work towards tool-support for UML [Ste00].

(2) Through its different kinds of diagrams, UML allows to take different views on the system (e.g. on the physical layer). This allows expressing properties of and reasoning about boundaries of critical components.

(3) For stepwise development and modular reasoning, one may make use of work in [Jür01d, Jür01a] giving a notion of secrecy preserved under refinement and giving compositionality results in the setting of a simple specification language (the current work aims to transport these results into the setting of UML).

(4) The question to what extent the formal methods approach to modelling cryptographic primitives is sound is addressed in [AJ01], where a translation from a symbolic semantics of a specification language involving cryptographic operations to a complexity-theoretical one is given. Again these results apply (after adjusting the differences in the models) in particular to a formal core of UML.

---

\(^1\) For some background on giving UML a formal semantics cf. e.g. [EFLR99].
(5) One may address vulnerabilities arising from flaws in the implementation by deriving test-sequences for implementations from formal specifications (e.g., following ideas from [JW01b], where this is done using a formal notation close to UML).

Beyond verification, one may also employ UML in requirements capture for security protocols [Low00] or to encapsulate rules of prudent engineering for secure systems [Jür01b].

In this talk we report on work towards verifying security properties in a formal core of UML in a compositional way.

References


