Model-based Security Engineering with UML: The Last Decade and Towards the Future

Jan Jürjens

http://jan.jurjens.de
Secure IT-Systems

Today IT-systems pervade almost all aspects of human life. At the same time, IT-systems become more open and therefore more vulnerable.

A lot of successful academic research has been done on foundations for secure systems. Some milestones:

• Saltzer, Schroeder: Protection of Information in Computer Systems, 1975
• Gasser: Building a Secure Computer System, 1988
• Burrows, Abadi, Needham: A Logic for authentication, 1989
• Ross Anderson: Security Engineering, 2001

Unfortunately, despite this successful research, today’s systems still often do not satisfy the increasing expectations on their security requirements - ...
How to Develop Secure IT-Systems?

... part of the problem is that:

1. Modern software engineering approaches in practice (which can manage today’s complex systems) usually do not consider security.

2. Traditional, practical approaches for security assurance do not provide a holistic, integrated assurance which would scale to the complexity modern systems in a reliable way.

To address this problem, 10 years ago a new line of research was started trying to bridge this gap by tailoring a modern software engineering approach (model-based development with UML) to the case of security-critical systems.
Model-based Security: Some Milestones

2001: UMLsec: UML profile for security modelling (Jürjens)
   Model-based security testing with AutoFocus (Wimmel, Jürjens)

2002: Secure UML: Modelling RBAC with UML (Basin et al.)
   Hypermedia security modeling with Ariadne (Aedo, Diaz et al.)
   Aspect-oriented Security Modelling (France et al.)
   Model-based IT security risk assessment (Stølen et al.)
   Interactive theorem proving of UML models for security (Haneberg, Reif et al.)

2003: Formal verification for UML models of access control (Koch, Parisi-Presicce)

2004: Automated verification tools for UMLsec (Shabalin et al.)
   Actor-centric modeling of user rights with UML (Breu et al.)
   Extending OCL for secure database development (Fernández-Medina et al.)

2005: First book on model-based security published (in English)

2007: Security monitors for UML policy models (Massacci et al.)

2008: Executable misuse cases for security concerns (Whittle et al.)

2009: Model-based security vs performance evaluation (Woodside et al.)
   First book on model-based security in Chinese

2010: From requirements to UMLsec models (Houmb et al.; Islam et al.; Mouratidis et al.)
   Security monitoring for UMLsec models (Bauer et al.; Pironti et al.)

...: Model-based security monitor generation for embedded systems (Schürr et al.)

Jan Jürjens  Model-based Security – UMLsec – Results – Challenges
Model-based Security: Some Tools

UMLsec Tool Framework (http://umlsec.de):
- Automated formal verification (model-checker, automated theorem prover) [Shabalin et al. 2004, 2005; Schreck et al. 2008]
- Code generation [Montrieux et al. 2010]

Secure UML (http://www.bm1software.com/eos)
- Currently no specific, openly available tool, but the OCL checker EOS can be used to check OCL annotations from SecureUML

CORAS (http://coras.sourceforge.net)
- Language editor for the CORAS notation

SECTET (http://qe-informatik.uibk.ac.at)
- Tool for configuring Security-as-a-Service architecture
Model-based Security: Industrial Usage
(some (published) examples)

2003: Internet bank architecture at HypoVereinsbank
   (Grünbauer et al.)
2005: Instant communication system (Apvrille et al.)
2007: Intranet information system at BMW (Best et al.)
2008: German Health Card architecture (Rumm et al.)
2008: Mobile security policies at O2 Germany (Bartmann et al.)
2009: Biometric authentication system (Lloyd et al.)
Model-based Security Engineering with UMLsec

Security Requirements
Integrate
Analyze

UMLsec Models

Code
Reverse Engin.
Code-/Testgen.

Generate
Verify
Monitor

Configuration Data

Configure

Runtime System

Execute

Jan Jürjens  Model-based Security – UMLsec – Results – Challenges
Security Requirements Engineering

Aims:

- Identify security requirements within the requirements elicitation.

Idea: “Requirements Mining” in security standards (e.g. Common Criteria) resp. in the given specification document

Validation example: IPTV Standard of Eur. Telecom. Stand. Inst. (ETSI)

Modeling with UMLsec

Aim:
- Documentation and automated analysis of security-relevant information (e.g. security properties and requirements) as part of the system specification.

Idea:
- UML for system modeling. [FASE 01, UML 02]
- Insert security-relevant information as stereotypes provided by UML-extension UMLsec.
- Formal semantics based on stream-processing functions as a foundation for verification. [Jour. Logic & Algebr. Program. '08]
Model-based Security Analysis

Aim:
- Automated analysis of the system models against the specified security requirements.

Idea:
Automated generation of logical formulas in first-order logic (or LTL, ...) based on formal semantics for security analysis. Transfer to the automatic theorem prover (or modelchecker/...).

[ICSE 05, ICSE 06]
**Model-based Security Testing**

**Problems** with using conformance-tests for security:

- In general, complete test coverage impracticable.
- Finds only attacks which are visible on the model level.

**Idea**: Mutation-testing.

- Focus on critical test cases
- Finds also weaknesses which are not visible on the model level.

**Validation**: Common Electronic Purse Specifications.
Detected several weaknesses.

[ASE 01, ICFEM 02]
Problem: Correct use of cryptography is inherently difficult to test: sufficient test coverage amounts to brute-force attack.

Idea: Automated, formal static program analysis of correct cryptographic function calls (with ATP for FOL).

Validation: Java Secure Sockets Extension (JSSE).

Current project Csec:
C code analysis.

[ICSM 05, ASE 05, ASE 06]
Security Analysis of Configuration Data

Aim: Verification if security policies are enforced by user permissions. Not feasible manually:

- Large amount of data (e.g. 60,000 permissions)
- Complex relations between permissions (e.g. delegation)

Idea: Automated analysis of business process models against user permissions, as well as user permissions against security policy models.

Current project (Fraunhofer Attract): Architecture for auditable business process execution (Apex).
Run-time Security Verification

**General problem:** Are verified implementations still secure in the system context?

- Does the static system model consider all relevant aspects?
- Are the assumptions about the system environment correct?
- Are the necessary abstractions for a static verification valid?

**Solution:** Run-time verification.

Classic approach: Fred Schneider's Security Automata (only safety properties).

New approach with 3-valued semantic for LTL: also non-safety properties.

**Validation with different versions** [Diss. A. Bauer]

Java Secure Sockets Extension.

Tool support

[UML 04, FASE 05, Jour. Softw. Tools & Techn. Transf. (STTT) 07]
General results: Security vs. Architecture

Idea: The model-based foundation allows one to investigate general questions regarding security preservation:

Security preservation vs. architectural principles:
- Horizontal layering of architectures
- Modularization / Composition of architectural components
- Service-oriented architectures
- Aspect-oriented architectures

Security preservation vs. development techniques:
- Refinement of specifications
- Refactoring of architectures

For each: Theorem providing conditions for preservation of security.

Jan Jürjens  Model-based Security – UMLsec – Results – Challenges
Security vs. Refinement

**Question**: Under which conditions does refinement (i.e. concretization of specifications) preserve security properties?

For behavior-preserving refinement one would expect security properties to be preserved.

„Refinement paradox“: In general not!

**Observation**: Problem: Mixture of nondeterminism for under-specification resp. as a security mechanism.

**Idea**: Separate the two on the modeling level.

**Theorem**: Then refinement preserves security requirements.

*Theorem*

- If $P$ preserves secrecy of $m$ and $P \sim Q$
  then $Q$ preserves secrecy of $m$.
Security vs. Modularization

**Idea:** Exploit architectural modularization for modular security verification.  

**Question:** Under which conditions does composition of components preserve security properties?  

Only works under suitable assumptions on other components.  

**Idea:** Formalize as „Rely-guarantee“-condition.  

⇒ Can verify components separately.  

**Validation:** Java Secure Sockets Extension.

---

**Theorem 5.** Let $P_1, P_2, D$ and $U$ be processes with $I_{P_1} = I_D$, $O_D = I_{P_2}$, $O_{P_2} = I_U$ and $O_U = O_{P_1}$ and such that $D$ has a left inverse $D'$ and $U$ a right inverse $U'$. Let 

$$m \in (\text{Secret} \cup \text{Keys}) \setminus \bigcup_{Q \in \{D', U\}} (S_Q \cup K_Q).$$

- If $P_1$ preserves the secrecy of $m$ and $P_1 \xrightarrow{(D,U)} P_2$ then $P_2$ preserves the secrecy of $m$. 

---
Validation Example: Internal information system at BMW

MetaSearch Engine: personalized search in corporate intranet (password-protected).

Some documents are very security-critical.

Over 1,000 potential users, 280,000 documents, 20,000 requests per day.

Seamlessly integrated into enterprise security architecture. Provides security services for applications (user authentication, role-based access, global single-sign-on), starting point for further security services.

Successfully analyzed with UMLsec.

[ICSE 07]
Further Applications

- German Health-card: Architecture analyzed with UMLsec, some weaknesses found [Jour. Meth. Inform. Medicine '08]
- Mobile security policies at O₂ (Germany) [ICSE '08]
- Internet bank architecture at HypoVereinsbank [SAFECOMP '03]
- Common Electronic Purse Specifications (Global standard for electronic purses): several weaknesses found [ASE '01]
- Biometric authentication systems: several weaknesses found [ACSAC '05, Models '09]
- Health information systems [Caise '09]
Some Empirical Results

Is model-based quality assurance worthwhile compared to classical QA techniques (e.g. testing)?

1) Static Analysis vs. Code Review: Industrial software at O2 (Germany) examined for errors. Result:
- Static-analysis only finds certain error classes, but very reliably.
- Most important aim: reduce “false positive”-rate.

2) Model-checking vs. Simulation / Tests: door control (in coop. w. BMW). Typical error classes:
- Simulation / testing finds many “simple” errors fast and effectively (e.g. incorrect transition priority: few min.)
- Model-checking also finds obscure errors (e.g. race conditions), but with additional effort (1-2 days for LTL formula).

[Testcom '05] [Models '08]
**UMLsec: Summary**

Model based security engineering with UMLsec:

- Model-based development with UML
- Automatic security analysis of software artifacts:
  - UML Models, Java / C programs, configuration data
- Successful applications in industry.

![Diagram showing the process of model-based security engineering with UMLsec](image)
Companies are increasingly active in Model-based Security
(e.g. Interactive Objects, ObjectSecurity, Thales (Security DSML), Foundstone (McAfee), …)


“Model-driven security is embryonic, but it will have a significant impact as information security infrastructures become increasingly real time, automated, and adaptive to changes in organizations and their environments.” [http://www.gartner.com/DisplayDocument?id=525109]

Note that everything said so far assumes that systems are built from scratch and won’t evolve, which of course is unrealistic…

Jan Jürjens  Model-based Security – UMLsec – Results – Challenges
The Forgotten End of the System Life-cycle

Challenges:

- Software lifetime often longer than intended (cf. Year-2000-Bug).
- Systems evolve during their lifetime.
- In practice evolution is difficult to handle [cf. HVB example].

Problem: Critical requirements (e.g. security) preserved?
Challenge: Evolution

Each artifact may evolve.

To reduce costs, reuse verification results as far as possible.

=> Under which conditions does evolution preserve security?

Even better: examine possible future evolution for effects on security.

• Check beforehand whether potential evolution will preserve security.
• Choose an architecture during the design phase which will support future evolution best wrt. security.

=> Evolution as first-class modeling concept in UMLsec.

Trade-off: flexibility of evolution vs. preservation of security.

[NB: analogous problem: Software-product lines.]
Secure Evolution: Tool support

Preserving requirements-code traceability by refactoring.

Jan Jürjens

Model-based Security – UMLsec – Results – Challenges
Model-based Security: Some Open Problems
(with a view to VL/HCC)

General challenges in model-based development
(particularly interesting / challenging / important when instantiated to security):

- **Scientific work:** model repositories? (in particular for security models)
- **Developer support:** model libraries? (in particular for security models)
  [NB: both exist for program source code!]
- **Industrial usage:** Studies on RoI of modelling?

Specific for model-based security:

- **Usability / scalability of security modelling notations / tools?**
  - How to represent complex information (such as security information) within visual diagrams in an understandable and usable way?
- **Human-centered security design?**
  - Use model-based security to evaluate security design alternatives wrt. usability (cf. passwords…)

Jan Jürjens  Model-based Security – UMLsec – Results – Challenges
Questions?

More information (papers, slides, tools etc.):
http://jan.jurjens.de