Methods of Safe and Secure Software Engineering

Jan Jürjens
Software & Systems Engineering
Informatics
TU Munich
tum

jan@jurjens.de
http://www.jurjens.de/jan
Critical Systems Development

High quality development of critical systems (safety-critical, security-critical,...) is difficult.

Many systems developed, fielded, used that do not satisfy their criticality requirements, sometimes with spectacular failures.

Correctness in conflict with cost. Thorough methods of system design not used if too expensive.
Model-based Development

Goal: ease the transition from human ideas to executed code.

Models
High-level languages
Machine code

Increase quality with bounded time-to-market and cost.
Using UML

UML: unprecedented opportunity for high-quality critical systems development feasible in industrial context:

• De-facto standard in industrial modeling: large number of developers trained in UML.
• Relatively precisely defined.
• Many tools in development.
Safety vs. Security

Safety = „Security against stupid adversaries“

Security = „Safety for paranoids“

Adversaries in security correspond to failures in safety.
UMLsafe/sec: goals

Extensions for safe/secure systems development.

- evaluate UML specifications for weaknesses in design
- encapsulate established rules of prudent safety/security engineering as checklist
- make available to developers not specialized in safety/security-critical systems
- consider safety/security from early design phases, in system context
- make certification cost-effective
The UMLsafe/sec profiles

Recurring safety/security requirements, failure/attack scenarios, concepts (fault tolerance/cryptography) offered as stereotypes with tags on component-level. Use associated constraints to evaluate specifications and indicate possible weaknesses.

Ensures that UML specification provides desired level of safety/security. Link to code via test-sequence generation.
<<data security>>

<<critical>> data security requirements enforced against threat scenario in deployment diagram.

Constraints:

Secrecy of \{secrecy\} data preserved.

Integrity of \{integrity\} data preserved.
Example <<data security>>

Variant of TLS (INFOCOM 1999).
Violates \{secrecy\} of s against default adversary.
Formal semantics

Diagrams in context (using subsystems). Model actions and internal activities explicitly.

Message exchange between objects or components (incl. event dispatching).

For UMLsec/safe: include adversary/failure model arising from threat/risk scenario in deployment diagram.

Use Abstract State Machines (pseudo-code).
Security Analysis

Model adversary (Dolev, Yao 1982).
Knows some data in advance.
May attack system parts in certain way (read, delete, insert data).
Example: insider attacker may intercept communication links in LAN.
To evaluate security of specification, execute jointly with adversary.
Formal vs. complexity-theoretical

Complexity-theoretical approach to cryptoprotocols.
• foundationaly more satisfying,
• more complete.

Formal methods-based approach
• closer to general methods for reasoning about reactive systems,
• easier to apply,
• easier to link to naive intuitions.
Bridging the gap

Soundness property (desired):
If security property proved formally, then holds in computational model (with high probability, for computationally reasonable attacks).

Formal proof will
- not mention probabilities and complexity,
- consider attacks only in the formal model,
- establish an all-or-nothing statement.

Abadi, Rogaway 2000: Soundness of single messages
Abadi, Jürjens 2001: Soundness of protocol runs with passive adversary.
Safety

Exchanged data may be
• delayed (and possibly reordered)
• lost
• corrupted.

Often, failures occur randomly (e.g. hardware).
Failure semantics examples:
• crash/performance: component may crash or exceed time limit, but partially correct.
• value: component may deliver incorrect values.

Include redundancy model.
Tool

- UML Editor (UML 1.4 / XMI 1.2 - compliant)
  e.g. Poseidon 1.6
- UML Model (UML 1.4 / XMI 1.2)
- Modified UML Model
- Text Report
- MDR
- JMI
- Error Analyzer
- Static Checker
- Dynamic Checker
- Analysis Suite
- SMV Model Checker
- SMV Model and CTL properties
- Counter-Example
Some resources

Book: Jan Jürjens, Secure Systems Development with UML, Springer-Verlag, due 2003

Tutorials: May: CSS (Mexico), Infotage (München); June: UMLws (CA); Sept: FME (Pisa), FDL (Frankfurt), SAFECOMP (Edinburgh)

More information: http://www4.in.tum.de/~secse