Developing Secure Systems with UMLsec: From Business Processes to Implementation

Jan Jürjens

Computing Laboratory, University of Oxford

(from 1 Oct: Technical University of Munich)

jan@comlab.ox.ac.uk
http://www.jurjens.de/jan

Motivation

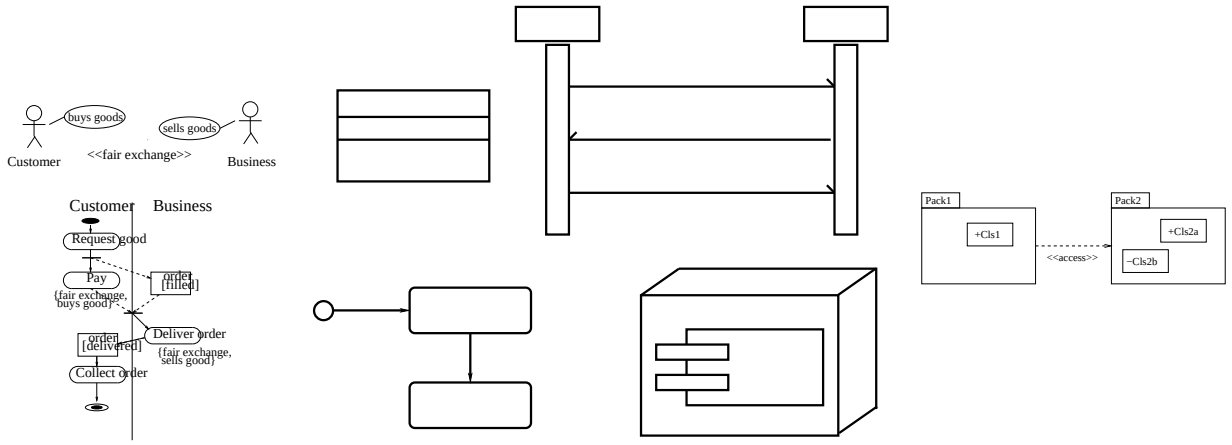
Computer security increasingly important (networks),
but designers often lack background in security.

Cannot use security mechanisms "blindly":
Security often compromised by circumventing
(rather than breaking) them.

Get security assurance into design process:

- Encourage and enable developers
  to consider security from early design phases.
- Encapsulate knowledge on prudent security engineering
  to aid secure systems development.
- Gain confidence on system security by verification.

UML



Unified Modeling Language (UML):
de-facto industry standard for object-oriented modelling.

Different kinds of diagrams for different views on the system.

Verification of behavioural properties: formal semantics.

UML diagrams

- Use case diagram: typical interaction between user and system
- Activity diagram: flow of control between system components
- Class diagram: class structure of the system
- Sequence diagram: interaction between components by message exchange
- Statechart diagram: dynamic component behaviour
- Package: collect system parts into groups
- Deployment diagram: Components in physical environment.

Distributed systems

Objects distributed over untrusted networks.

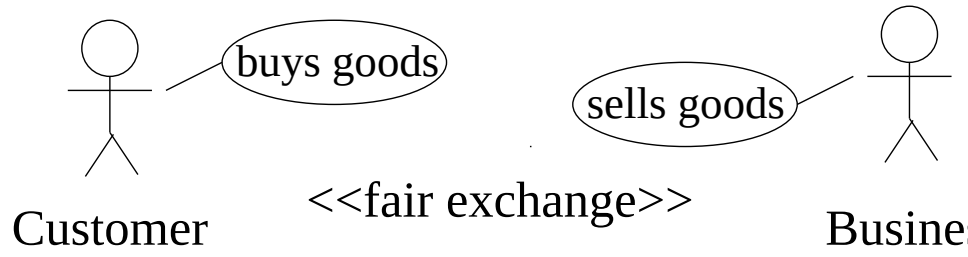
"Adversary" intercepts, modifies, deletes, inserts messages.

Cryptographic protocols to exchange session keys etc.

Vulnerabilities often at boundary between protocols and system.

Protocols in the context of system development with UML.

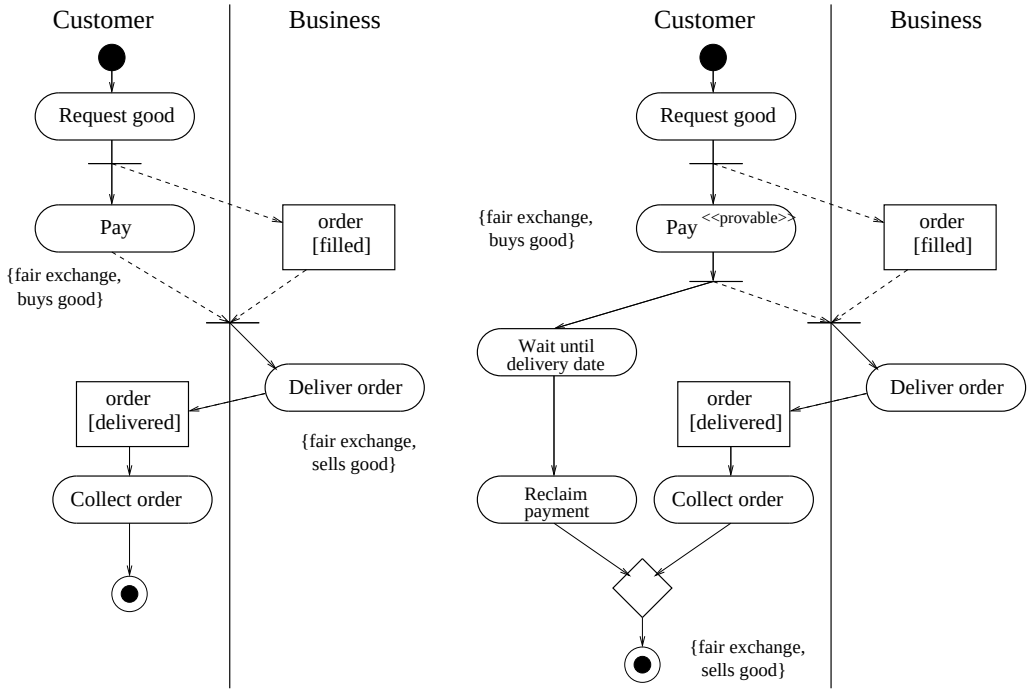
Use case diagrams: Security requirements



Formulate security requirements on use cases as 《*stereotypes*》.

《*fair exchange*》: if "buys good" then eventually "sells good"

Activity diagrams: Secure control flow

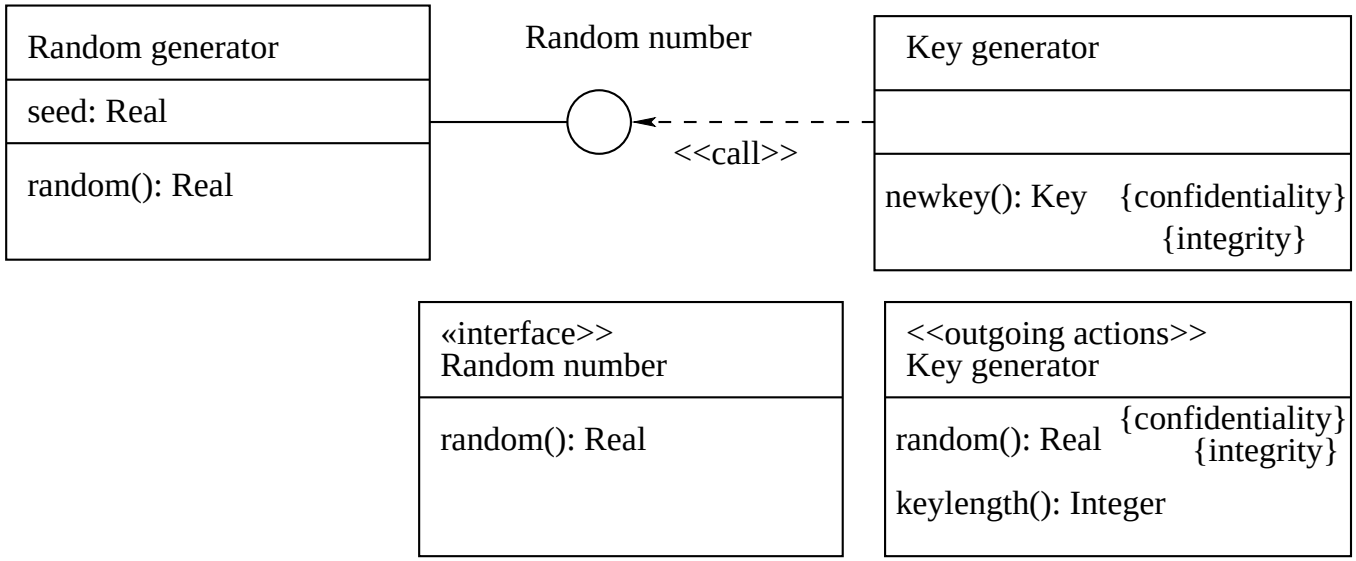


Ensure secure overall control flow (e.g. work-flow)

Second solution gives fair exchange if payment ⟪*provable*⟫

Proof using formal semantics

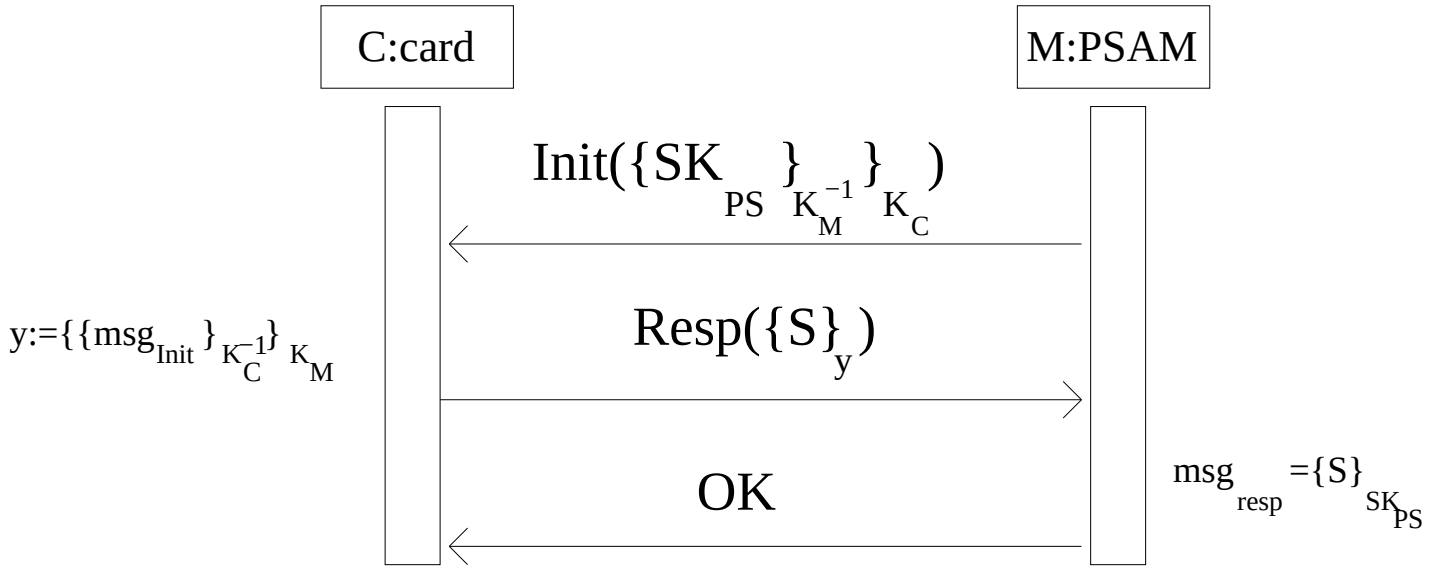
Class diagrams: Structural data security



Ensure class structure provides data security.

Operation random() does not guarantee required security.

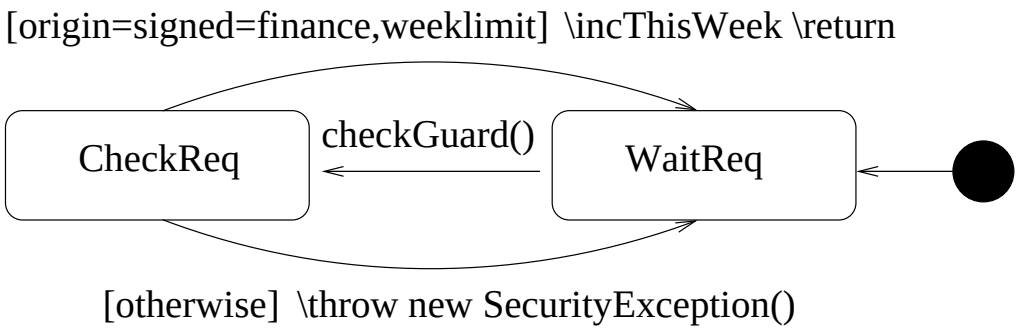
Sequence diagrams: Secure interactions



Use sequence diagrams to specify security protocols.

Translate to formal semantics; model-check or reason formally

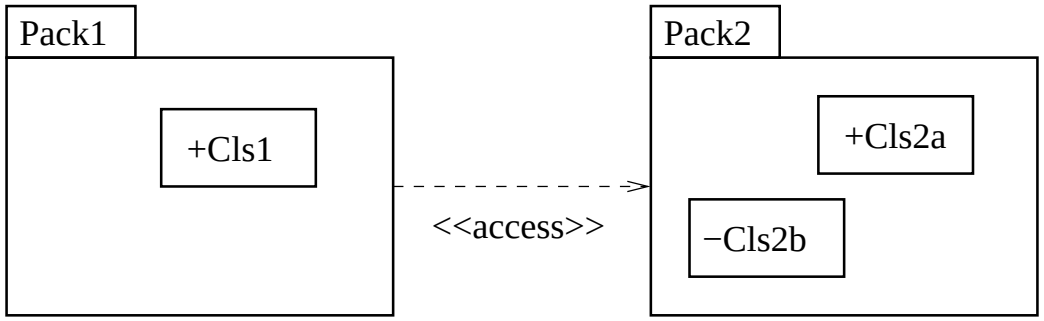
Statechart diagrams: Secure internal behaviour



Ensure secure behaviour within components/objects.

Access control, database security, information flow, . . .

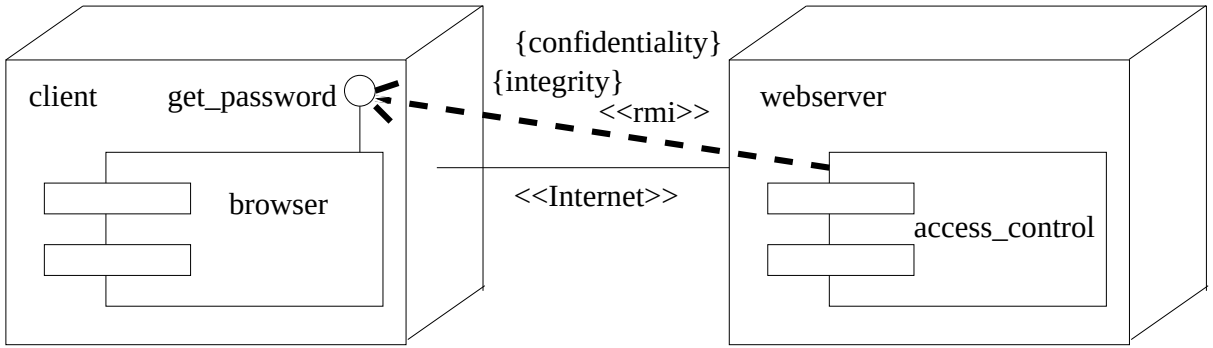
Package diagrams: Secure components



Component-based reasoning using package diagrams.

Uses visibility of parts within packages.

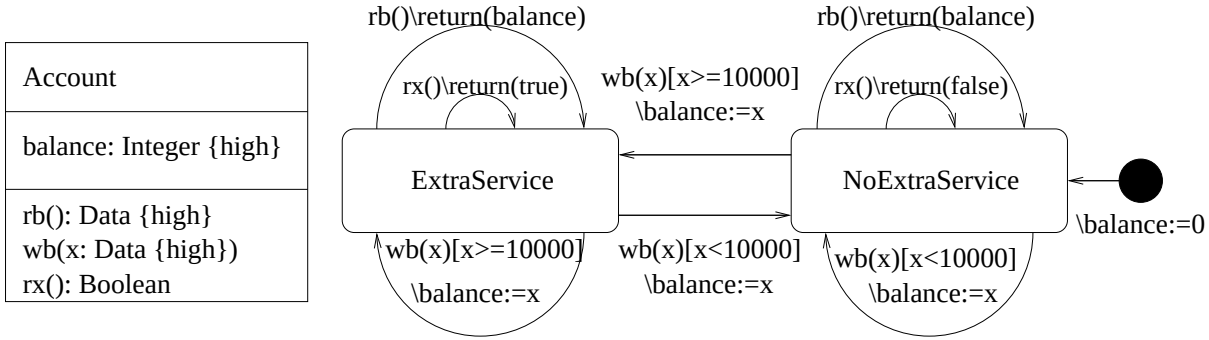
Deployment diagrams: Physical layer



Express security assumptions on physical layer of the system (security of communication links, hardware security, tamper resistance . . .)

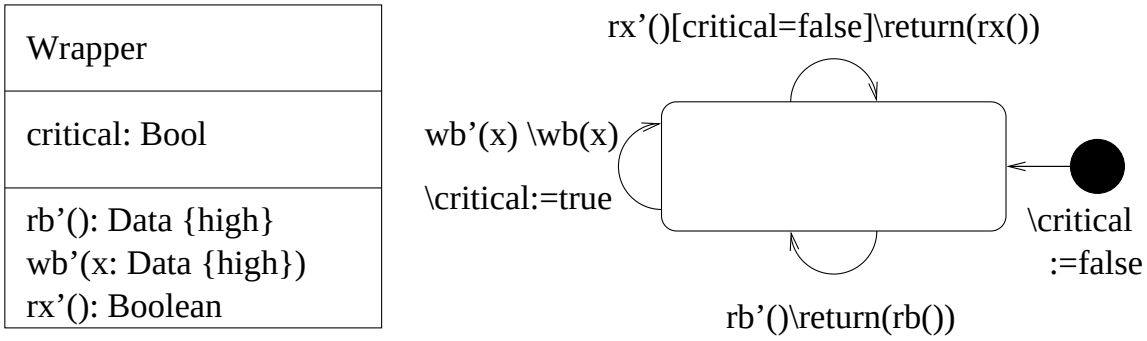
Security Patterns

Security patterns: use UML to encapsulate knowledge of prudent security engineering



Database object does not preserve security of account balance.

Wrapper Pattern



Use wrapper pattern to ensure that no low read after high write.

Related Work

Formal methods for computer security
(Burrows, Abadi, Needham; Meadows et al.; Roscoe, Lowe, ...)

Security design principles (Saltzer, Schroeder;
Abadi, Needham, Anderson)

Security and software engineering (Devanbu, Fong, Stubblebine)

Security patterns (Fernandez et al.)

Conclusion

Promising step towards secure systems development with UML.

Security by design:

- start in early design phases
- encapsulate security knowledge
- make verification more usable in practice
- reduce cost of official evaluation (reuse existing UML specs)

Further work

- developing secure Java systems using UML
  (signing/sealing/guarding objects...)
- modelling and analysis of
  Common Eletronic Purse Specifications with UML
- test case generation (using Autofocus, with G. Wimmel)

Future work

- more case studies
- tool support (e.g. statecharts simulator, with A. Cavarra)