

Using UMLsec and Goal Trees for Secure Systems Development

Jan Jürjens

Software & Systems Engineering
Informatics, TU Munich
Germany



juerjens@in.tum.de

<http://www.jurjens.de/jan>

Motivation

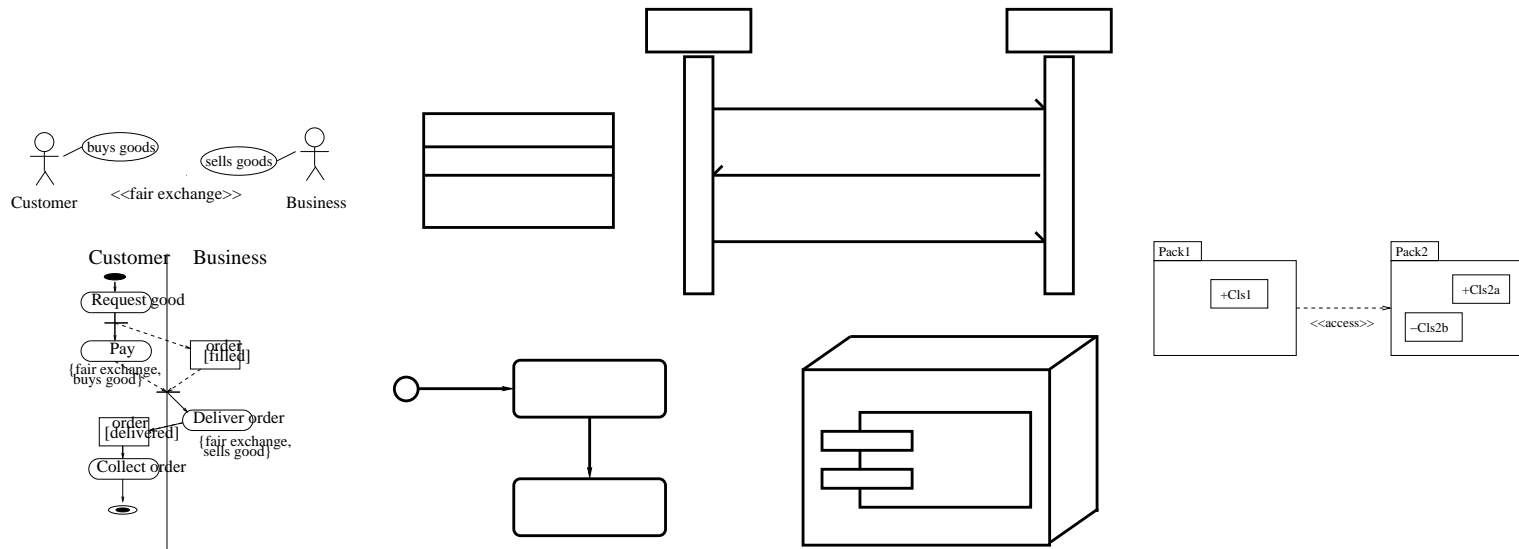
Computer security increasingly important (networks),
but designers often lack background in security.

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.

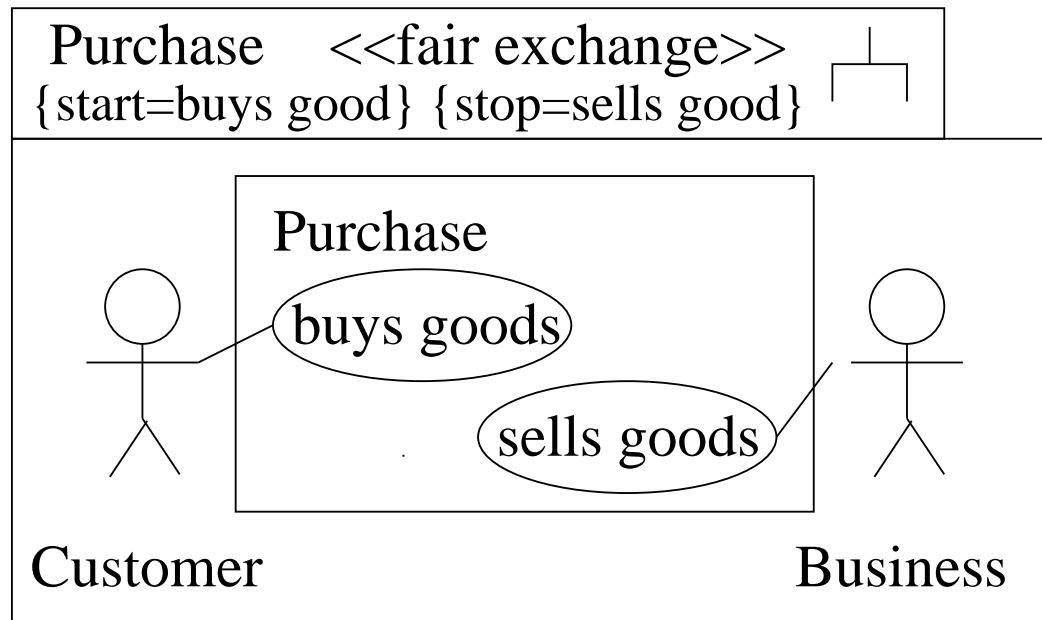
“Adversary” **intercepts, modifies, deletes, inserts** messages.

Cryptographic protocols to exchange session keys etc.

Vulnerabilities often at **boundary** between protocols and system.

Protocols in the context of system development with UML.

Requirements capture (use cases)



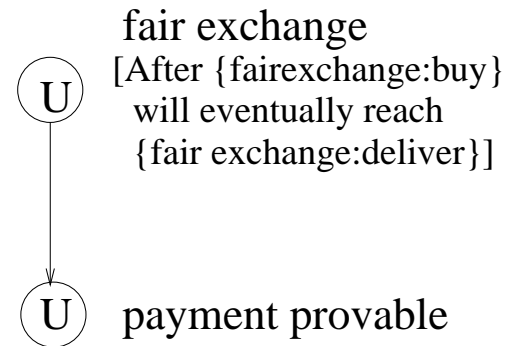
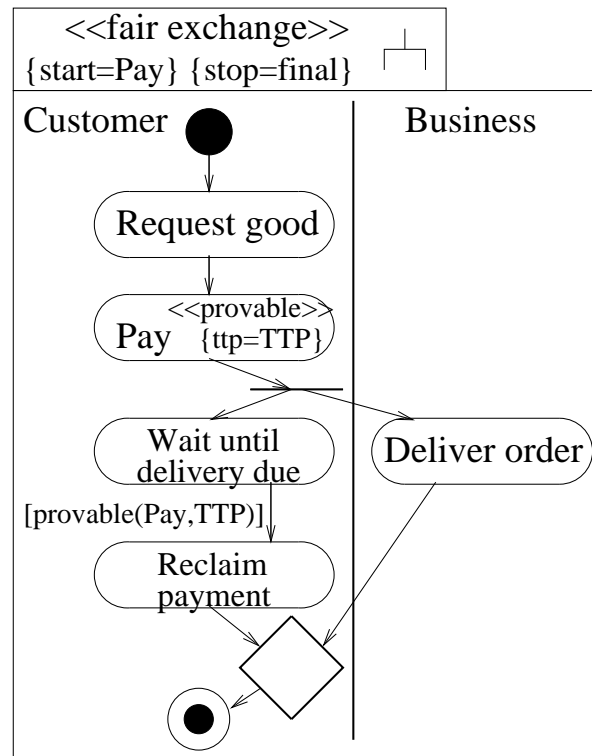
⊙ fair exchange

Goal tree

Formulate security requirements on use cases as
« stereotypes ».

« fairexchange »: if “buys good” then eventually “sells good”.

Analysis (activity diagrams)



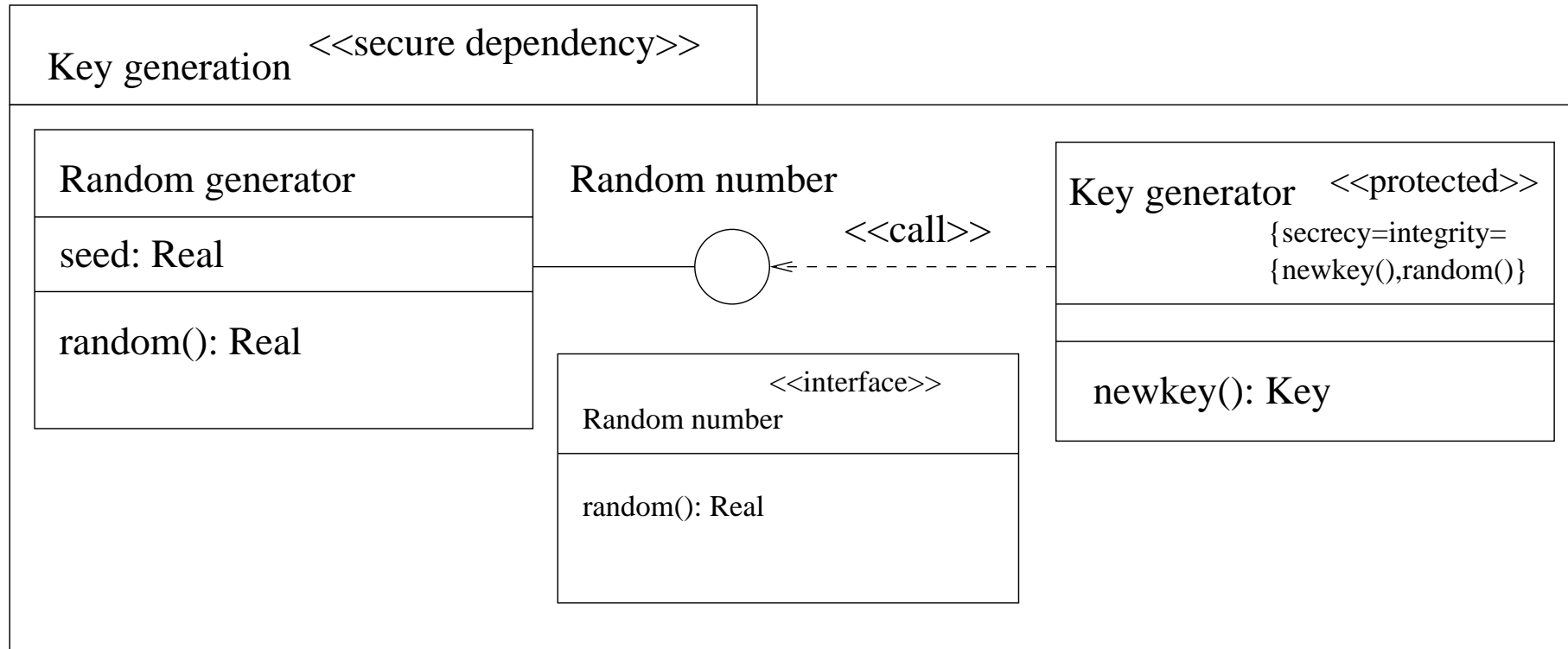
Goal tree

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

Gives fair exchange if payment « provable ».

Proof using formal semantics.

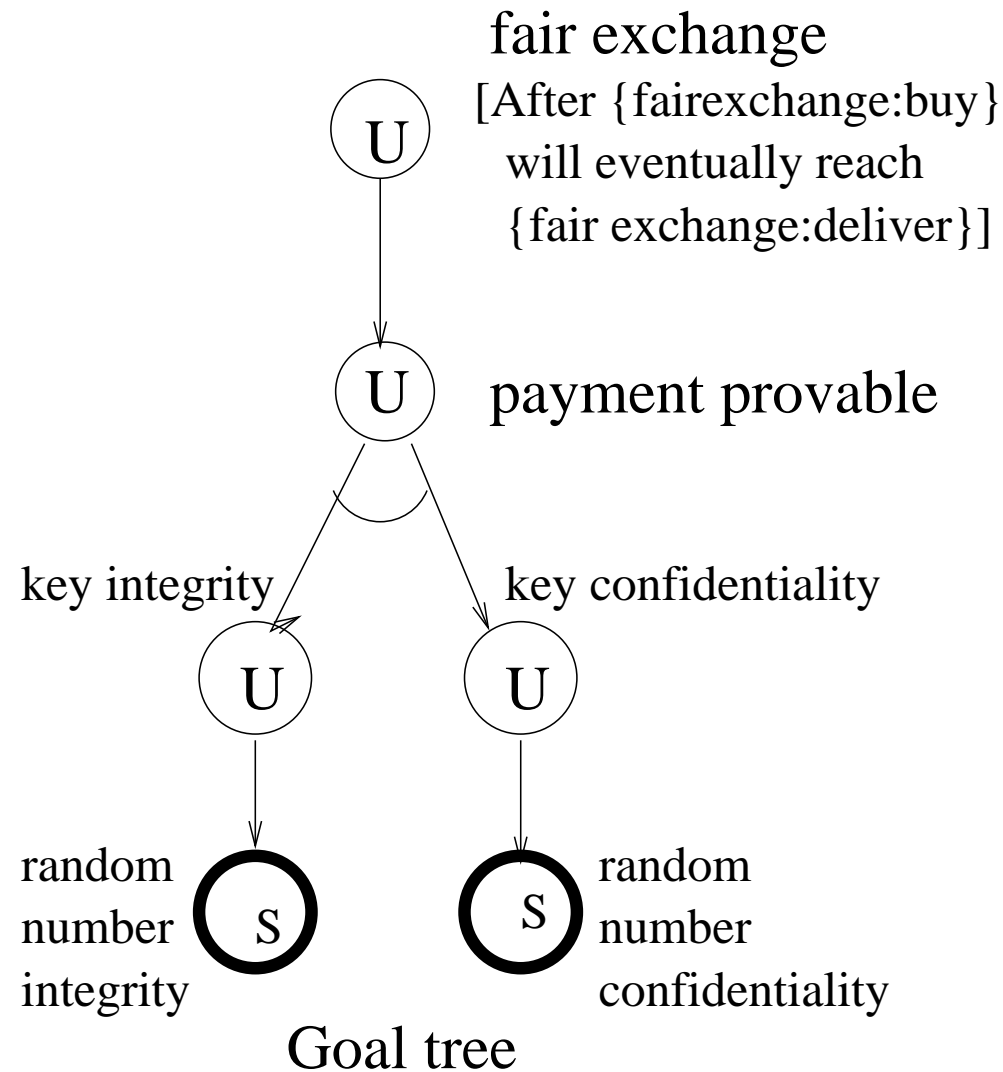
Design (class diagrams)



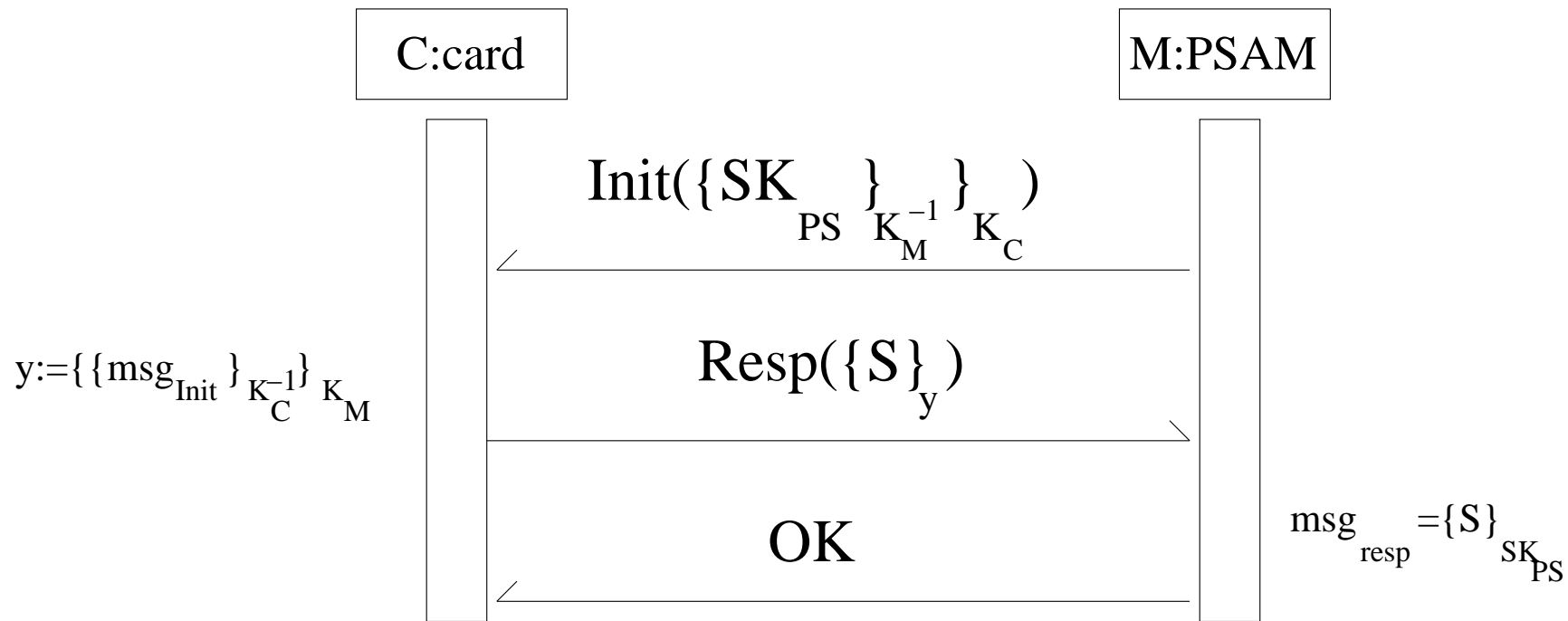
Ensure class structure provides data security.

Operation `random()` does not guarantee required security.

Design (class diagrams): Goal tree



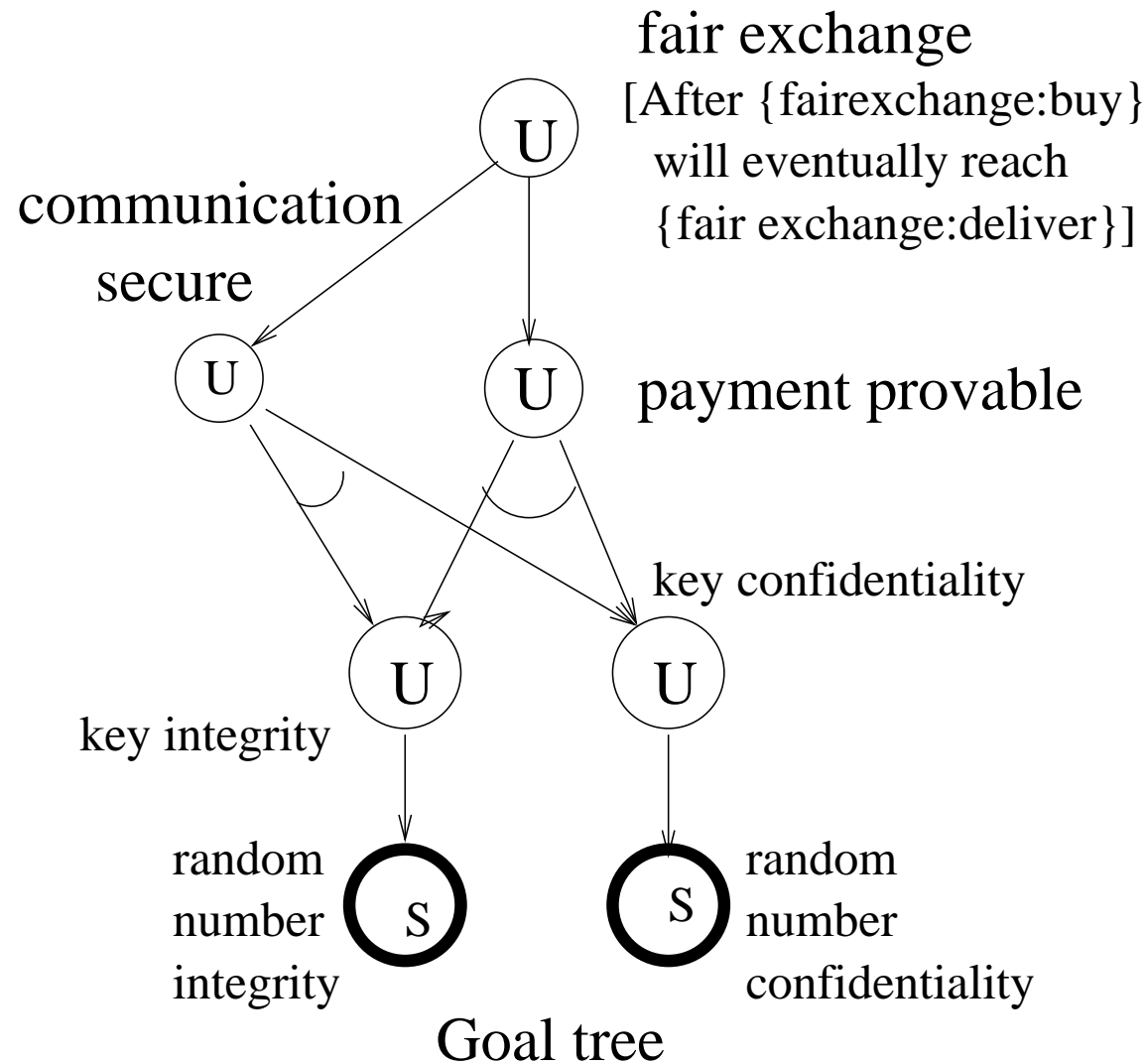
Design (sequence diagrams)



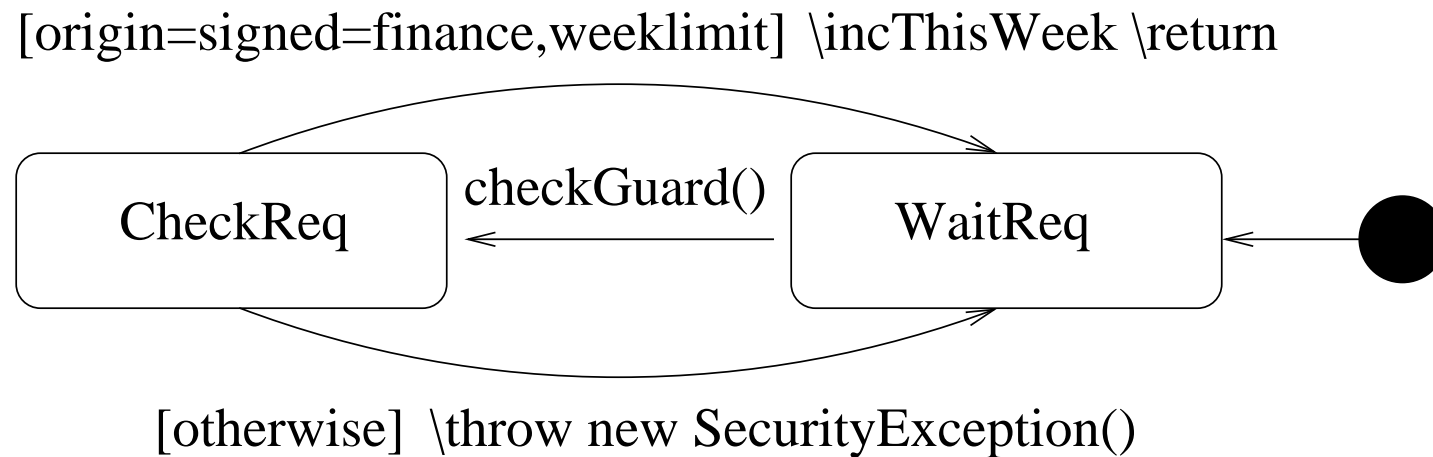
Use sequence diagrams to specify security protocols.

Translate to formal semantics; model-check or reason formally.

Design (sequence diagrams): Goal tree



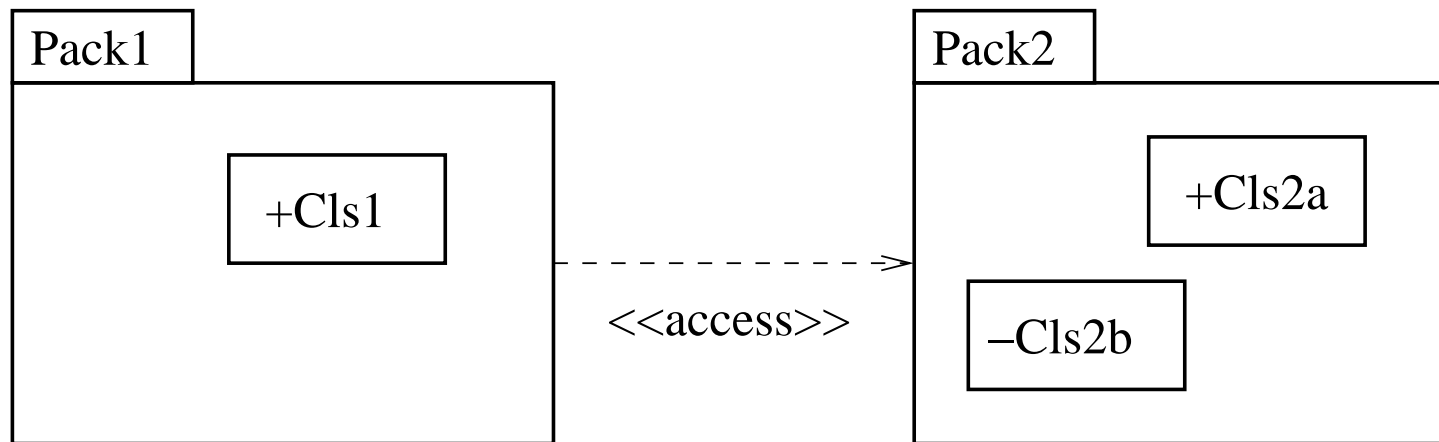
Design: statechart diagrams



Ensure secure behaviour **within** components/objects.

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

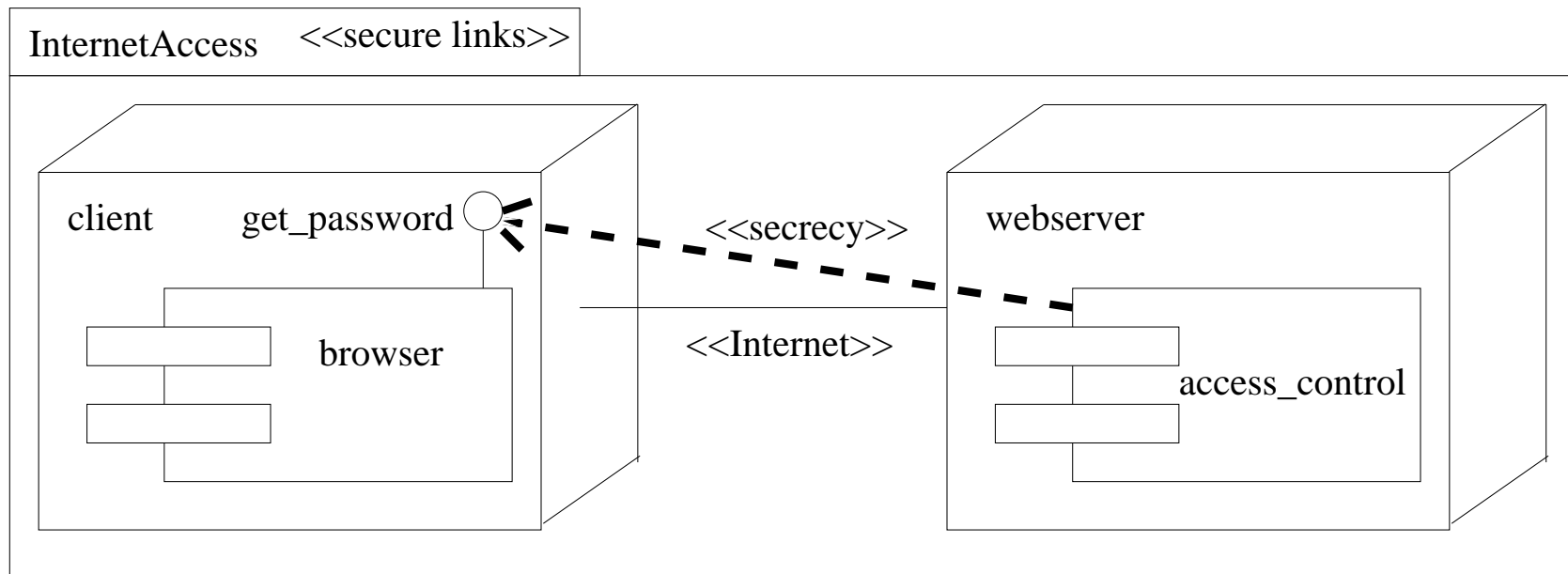
Design: package diagrams



Component-based reasoning using package diagrams.

Uses visibility of parts within packages.

Implementation: deployment diagrams



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

Conclusion

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 certification (reuse UML specs)

Combination of **use-case driven** and **goal-directed** process.

Further work

- developing secure Java systems using UML (signing/sealing/guarding objects...)
- modelling and analysis of Common Electronic Purse Specifications with UML

Future work

- tool support (UML tool via XMI to *AUTOFOCUS*)

Resources

Slides, papers etc.:

<http://www4.in.tum.de/~umlsec>

Homepage:

<http://www.jurjens.de/jan>

Thanks for your attention !