

# Secure Systems Development with UML: Applications to Telemedicine

Jan Jürjens

Software & Systems Engineering  
Informatics, Munich University of Technology  
Germany



[juerjens@in.tum.de](mailto:juerjens@in.tum.de)

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



# A need for Security

---

Society and economies rely on **computer networks** for communication, finance, energy distribution, transportation...

Attacks threaten **economical** and **physical** well-being of people and organizations.

Interconnected systems can be attacked **anonymously** and from a safe **distance**.

Networked computers need to be **secure**.

# Telemedicine

---

Telemedicine particularly critical:

- Human **life/health** (telesurgery)
  - **availability, integrity**
- Big **money** (medical (**genetic** !)) data vs. insurance companies)
  - **secrecy**

# Problems

---

Many **flaws** found in designs of security-critical systems, sometimes years after publication or use.

Example (1997):

NSA hacker team breaks into U.S.

Department of Defense computers and the U.S. electric power grid system. Simulates power outages and 911 emergency telephone overloads in Washington, D.C..

# Causes I

---

- Designing secure systems correctly is **difficult**.
- Designers often **lack** background in security
- Security as an **afterthought**.

Even experts may fail:

- Needham-schroeder protocol (**1978**)
- attacks found **1981** (Denning sacco),  
**1995**(Lowe)

# Causes II

---

Cannot use security mechanisms „blindly“:

- Security often compromised by **circumventing** (rather than **breaking**) them.
- Assumptions on system **context**, physical environment.

„Those who think that their problem can be solved by simply applying cryptography don't understand cryptography and don't understand their problem“ (Lampson/Needham).

# Difficulties

---

Exploit information spreads **quickly**.

**No feedback** on delivered security from customers.

# Previous approaches

---

„Penetrate-and-patch“: unsatisfactory.

- **insecure** (damage until discovered)
- **disruptive** (distributing patches **costs** money, **destroys** confidence, **annoys** customers)

Traditional formal methods: **expensive**.

- training people
- constructing formal specifications.

# Goal: Security by design

---

Consider security

- from **early** on
- within **development** context
- in a seamless way.

„An **expansive** view of the problem is most appropriate to help ensure that no gaps appear in the strategy“ (Saltzer, Schroeder 1975).

But „no complete methode applicable to the construction of large general-purpose systems exists yet“ -since **1975**.

# Quality vs. cost

---

**Correctness** in conflict with **cost**.

Thorough methods of system design  
not used if too **expensive**.

# 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 (given the user community).
- Many **tools** in development (also for analysis, testing, simulation, transformation ).

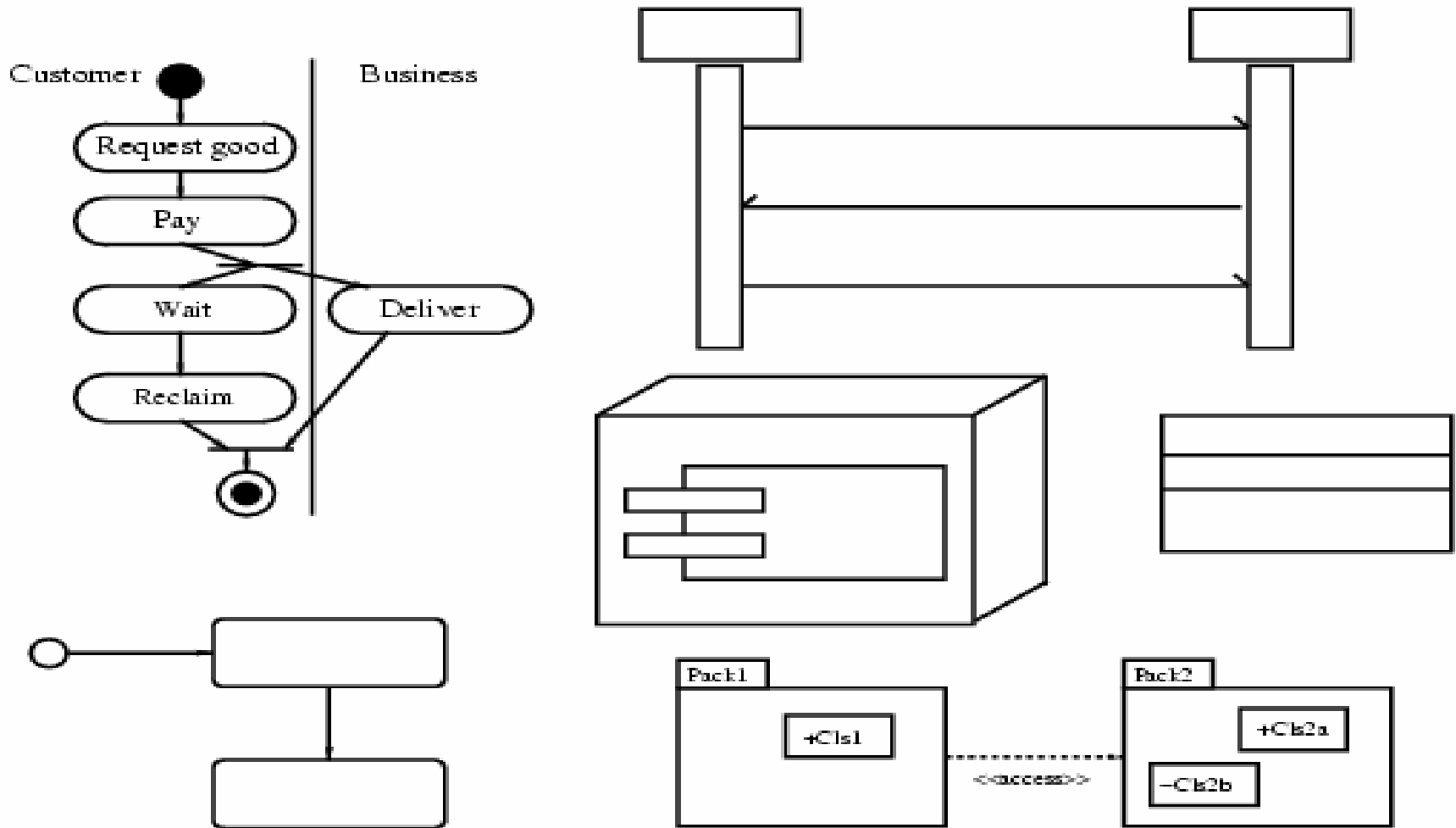
# Using UML

---

Unified modeling Language (UML):

- **visual** modeling for OO systems
- different **views** on a system
- high degree of **abstraction** possible
- de-facto industry **standard** (OMG)
- standard **extension** mechanisms

# A glimpse at UML



# Used fragment of UML

---

**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

**Deployment diagram:** Components in physical **environment**

**Package:** **collect** system parts into groups

Current: UML 1.4 (released Feb. 2001)

# UML Extension mechanisms

---

Stereotype: **specialize** model element using `<<label>>`

Tagged value: **attach** `{tag =value}` pair to stereotyped element

Constraint: **refine** semantics of stereotyped element

Profile: **gather** above information

# UMLsec

---

UMLsec: extension for **secure systems** development.

Goals :

- evaluate UML specifications for **vulnerabilities** in design
- encapsulate **established rules** of prudent security engineering
- make available to developers **not specialized** in security
- consider security from **early** design phases, in system **context**
- make certification **cost-effective**

# The UMLsec profile

---

**Recurring** security requirements offered as stereotypes with tags (secrecy, integrity,...).

Use associated constraints to **evaluate** specifications and indicate possible **vulnerabilities**.

Ensures that stated security requirements **enforce** given security policy.

Ensures that UML specification **provides** requirements.

# UMLsec profile (excerpt)

Stereotype	Base class	Tags	Constraints	Description
Internet secure links	link subsystem		dependency security matched by links	Internet connection enforces secure communic. links
secrecy secure dependency	dependency subsystem		call, send respect data security	assumes secrecy structural data security
no down-flow data security	subsystem subsystem	high	prevents down-flow provides secrecy integrity	information flow basic datasec requirements
fair exchange	package	start, stop	after start eventually reach stop	enforce fair exchange

<<Internet>>, <<encrypted >>, <<LAN >>, ...

---

Denote kinds of communication **links** resp. system **nodes**.

For adversary type  $A$ , stereotype  $s$ , have set  $\text{Threats}_A(s) \in \{\text{delete, read, insert, access}\}$  of actions that adversaries are capable of.

Default attacker:

Stereotype	$\text{Threats}_{\text{default}}()$
Internet	{delete, read, insert}
encrypted	{delete}
LAN	$\emptyset$
smart cart	$\emptyset$
POS device	$\emptyset$

# <<secure dependency>>

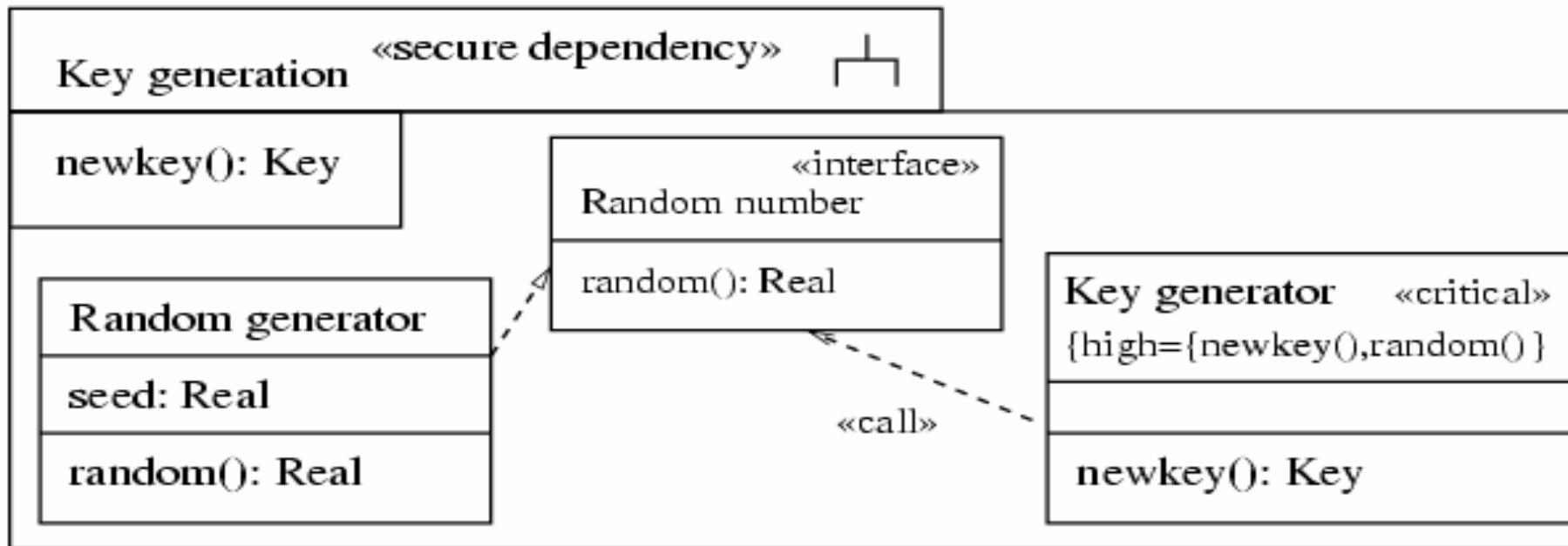
---

Ensures that <<call>> and <<send>> dependencies between components **respect** security requirements on communicated data given by tag {high}.

Constraint: given <<call>> or <<send>> dependency from *C* to *D*:

- Any message *n* in *D* appears in {high} in *C* if and only if it does so in *D*.
- If message in *D* appears in {high} in *C*, corresp. dependency stereotyped <<high>>.

# Example <<secure dependency>>



Specification **violates** constraint for <<secure dependency>>: Random generator and <<call>> dependency do not provide security levels for random() required by key generator.

<<no down-flow>>, <<no up flow>>

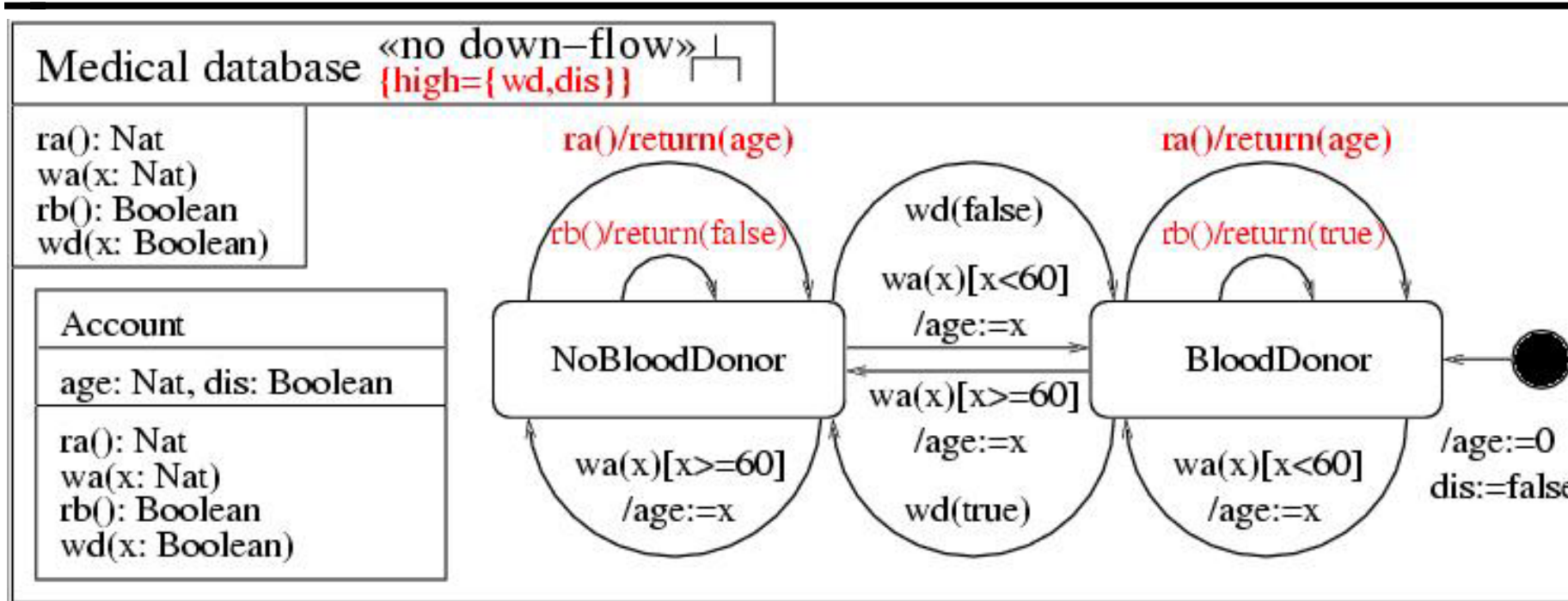
---

Enforce secure **information flow**. Constraints:

<<no down-flow>>: component prevents down flow: Value of any data specified in {high} may influence **only** the values of data also specified in {high}.

<<no up-flow>>: component prevents up-flow: Value of any data specified in {high} may be influenced **only** by the values of data also specified in {high}.

# Example <<no down-flow>>



Database object allows age, fitness to be blood donor to be read. Does **not** provide <<no down-flow>>: derive partial information about dis.

# Formal semantics for UML: Why

---

Meaning of diagrams stated **imprecisely** in (OMG 2001).

**Ambiguities** problem for

- tool support
- establishing behavioral properties (e.g. security)

Need **precise** semantics for used part of UML, especially to ensure security requirements.

# Formal semantics for UML: How

---

Diagrams in **context** (using subsystems).

Model **actions** and internal **activities** explicitly.

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

For UMLsec: include **adversary** arising from threat scenario in deployment diagram.

Use Abstract State Machines (pseudo-code).

# Distributed Systems

---

Objects distributed over **untrusted** networks.

„Adversary“ intercepts, modifies, deletes, inserts messages.

**Cryptography** provides security.

# Security Analysis

---

Model classes of **adversaries**.

May attack different parts of the system in a specified way.

Example: **insider** attacker may intercept communication links in LAN.

To evaluate security of specification, execute jointly with adversary.

# Risk Analysis

---

- Within adversary model: attach probabilities to failures (e.g. compromise of key)
- Derive probabilities for hazards

# Connection with analysis tool

---

Commercial modelling tools: only **syntactic** checks and **code-generation**.

Current work: connect to **verification** tools.

Industrial CASE tool with UML-like notation:

**AUTOFOCUS**

(<http://autofocus.informatik.tu-muenchen.de>)

- verification
- code generation
- test-sequence generation

# Conclusion

---

- Use UML extension **UMLsec** for model-based development of security-critical systems
- Apply to telemedicine systems
- Work-in-progress: mechanical tool-support

# Resources

---

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

Satellite workshop at UML'02 (Dresden/Germany, 30 Sep-4 Oct) on Critical Systems Development with UML

More information (also slides, papers etc.):

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

**Thanks for your attention !**