Developing Safety- and Security-critical Systems with UML

Jan Jürjens

Software & Systems Engineering

Informatics, Munich University of Technology

Germany

TUM

juerjens@in.tum.de

http://www.jurjens.de/jan

TUM

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: easen transition from human ideas to executed code.

Models

High-level languages

Machine code

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

Using UML: Why

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

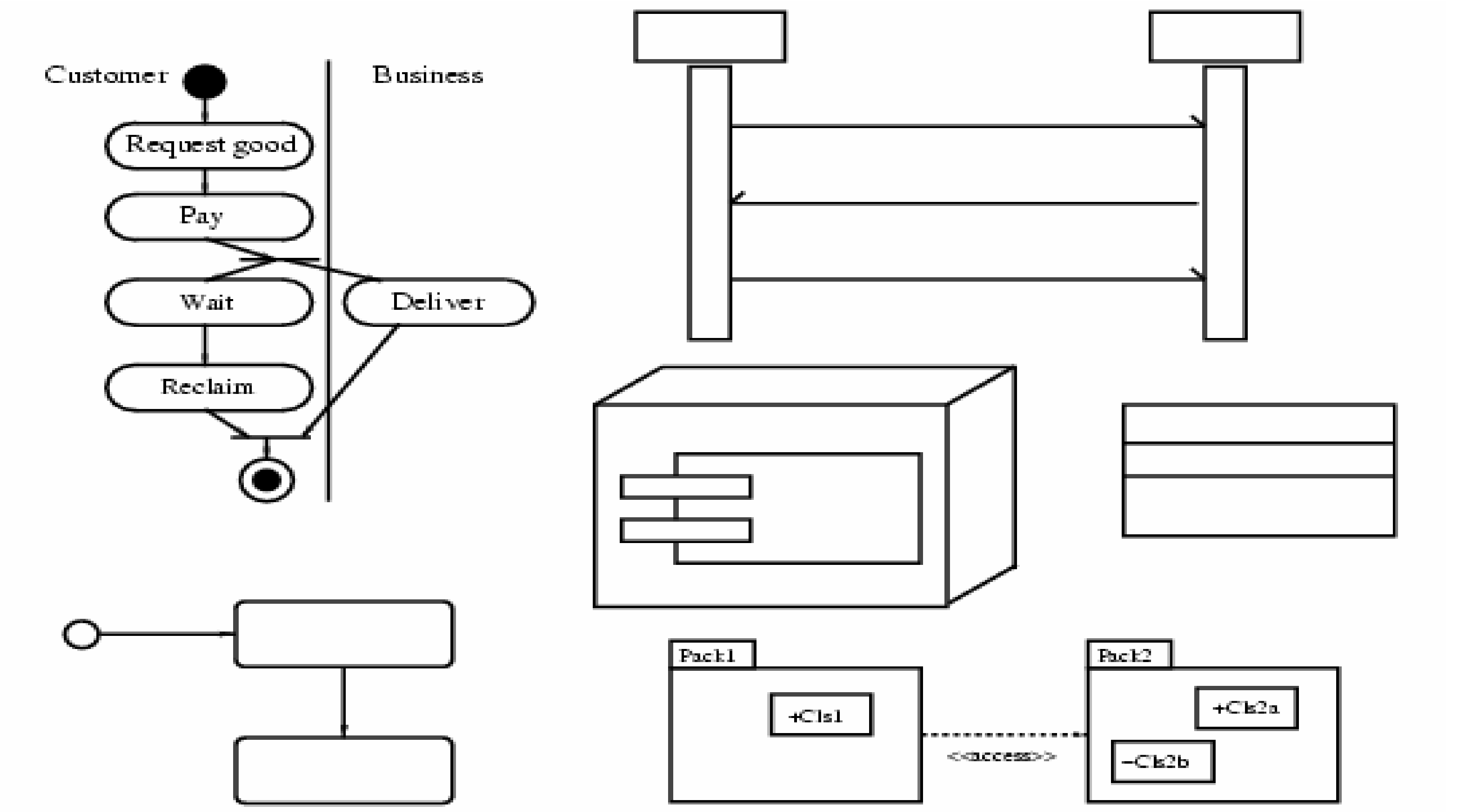
- De-facto standard in industrial modeling: large number of developers trained in UML.
- Relatively precisely defined (given the user community).
- Many tools in development (also for analysis, testing, simulation, transformation).

Using UML: What

Unified Modeling Language (UML):

- visual modelling 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 ¿labelÀ.

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

Constraint: refine semantics of stereotyped element.

Profile: gather above information.

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.

Safety

Safety-critical systems: five failure condition categories: catastrophic, hazardous, major, minor, no effect.

Corresponding safety levels A - E (DO-178B standards in avionics).

Safety goals: via the maximum allowed failure rate. For high degree of safety, testing not sufficient (1 failure per 100,000 years).

Failures

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.

Fault-tolerance

Redundancy model determines which level of redundancy provided.

Goal: no hazards in presence of single-point failures.

Failure semantics modelling

For redundancy model *R*, stereotype $s \in$ {¿crash/performanceÀ, ¿valueÀ}, have set $Failures_R(s) \subseteq$ {delay(*t*), loss(*p*), corrupt(*q*)}:

- *t*: expected maximum time delay,
- *p*: probability that value not delivered within *t*,
- *q*: probability that value delivered in time corrupted

(in each case incorporating redundancy). Or use ¿riskÀ stereotype with {failure} tag.

Example

Suppose redundancy model *R* uses controller with redundancy *3* and the fastest result. Then could take:

- delay(*t*): *t* delay of fastest controller,
- loss(*p*): *p* probability that fastest result delivered within *t*,
- loss(*p*): *p* probability that fastest result is corrupted.

Failure models

lq^l_n: messages on link l delayed further n time units.

p^h_n: probability of failure at n^{th} iteration in history h.

For link l stereotyped s where loss(p)2Failures$_R$(s),

- history may give lq^l_0:=;; then append p to $(p^h_n)_{n2N}$,
- or no change, then append 1-p.

For link l stereotyped s where corruption(q)2Failures$_R$(s),

- history may give lq^l_0:={¥}; then append q,
- or no change; append 1-q.

For link l stereotyped s with delay(t)2Failures$_R$(s), and lq^l_0≠;, history may give lq^l_n:=lq^l_0 for $n \cdot t$; append $1/t$.

Then for each n, lq^l_n:=lq^l_{n+1}.

Execution semantics

Behavioral interpretation of a UML subsystem:

(1) Takes input events.

(2) Events distributed from input and link queues between subcomponents to intended recipients where they are processed.

(3) Output distributed to link or output queues.

(4) Failure model applied as defined above.

¿guaranteeÀ

Describe guarantees required from communication dependencies resp. system components.

Tags: {goal} with value subset of {immediate(t), eventual(p), correct(q)}, where

- *t*: expected maximum time delay,
- *p*: probability that value is delivered within *t*,
- *q*: probability that value delivered in time not corrupted.

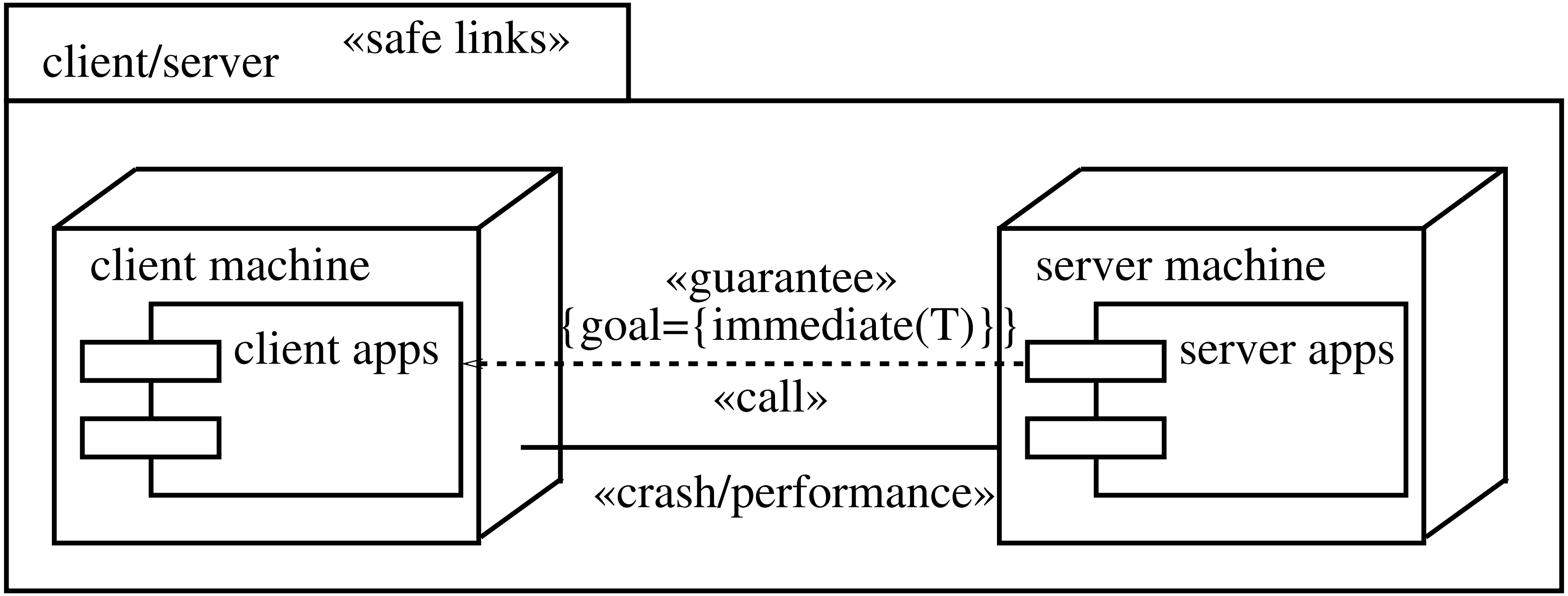
¿safe linksÀ

Physical layer should meet safety requirements on communication given redundancy model *R*.

Constraint: For dependency *d* stereotyped ¿guaranteeÀ have corresponding communication link *l* with stereotype *s* such that

- if {goal} has immediate(*t*) as value then delay(*t*‘)2Failures$_R$(*s*) implies *t*‘·*t*,
- if {goal} has eventual(*p*) as value then loss(*p*‘)2Failures$_R$(*s*) implies *p*‘·1-*p*, and
- if {goal} has correct(*q*) as value then corruption(*q*‘)2Failures$_R$(*s*) implies *q*‘·1-*q*.

Example ¿safe linksÀ



Given redundany model none, ¿safe linksÀ fulfilled iff T· expected delay according to Failures$_{none}$(¿crash/performanceÀ).

¿safe dependencyÀ

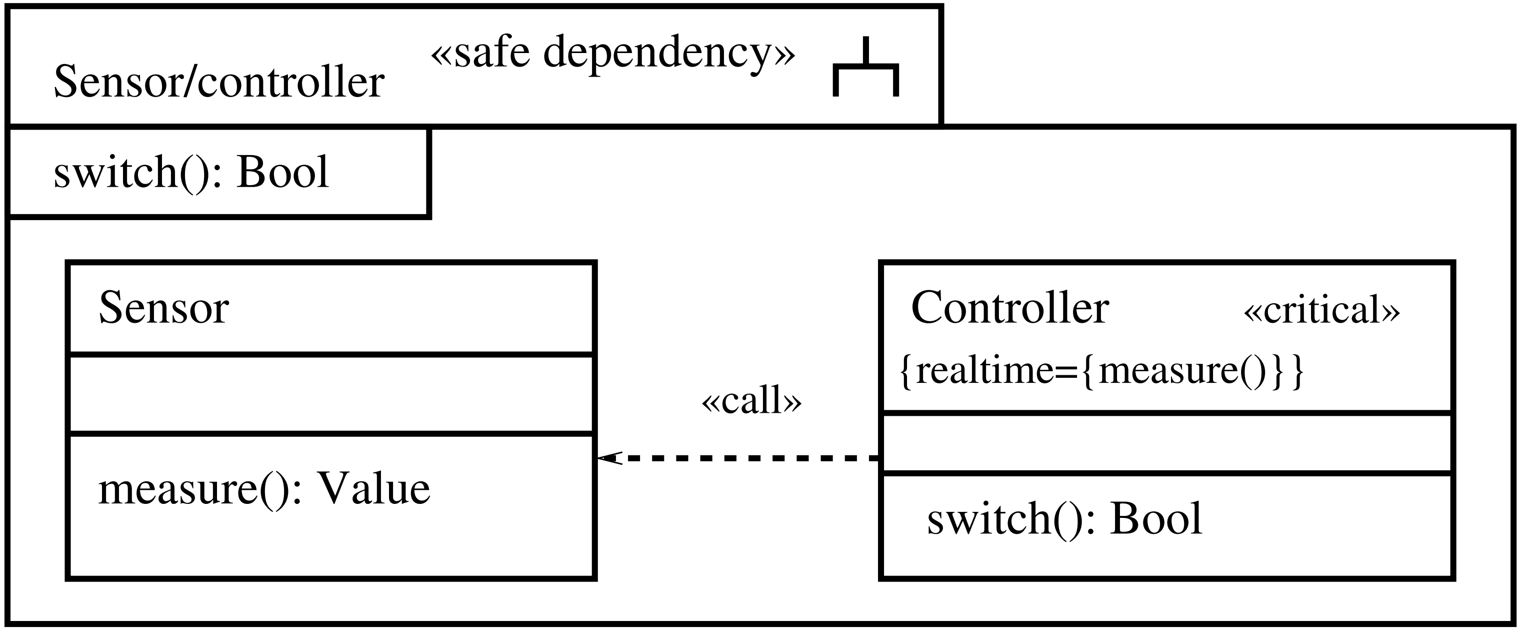
Communication dependencies should respect safety requirements on ¿criticalÀ data.

For each safety level {*l*} for ¿criticalÀ data, have goals(*l*)µ{immediate(*t*), eventual(*p*), correct(*q*)}.

Constraint: for each dependency *d* from *C* to *D* stereotyped ¿guaranteeÀ:

- Goals on data in *D* imply those in *C*.
- Goals on data in *C* also appearing in *D* met by guarantees of *d*.

Example ¿safe dependencyÀ



Assuming immediate(*t*)2goals(realtime), violates ¿safe dependencyÀ, since Sensor and dependency do not provide realtime goal immediate(*t*) for measure() required by Controller.

¿safe behaviourÀ

Ensures that system behavior in presence of failure model provides required safety {goals} by requiring that in any trace *h* of the execution:

- immediate(*t*): Value delivered after at most *t* time steps.
- eventual(*p*): Probability that delivered value is lost during transmission at most 1-*p*.
- correct(*q*): Probability that delivered value corrupted during transmission at most 1-*q*.

¿containmentÀ

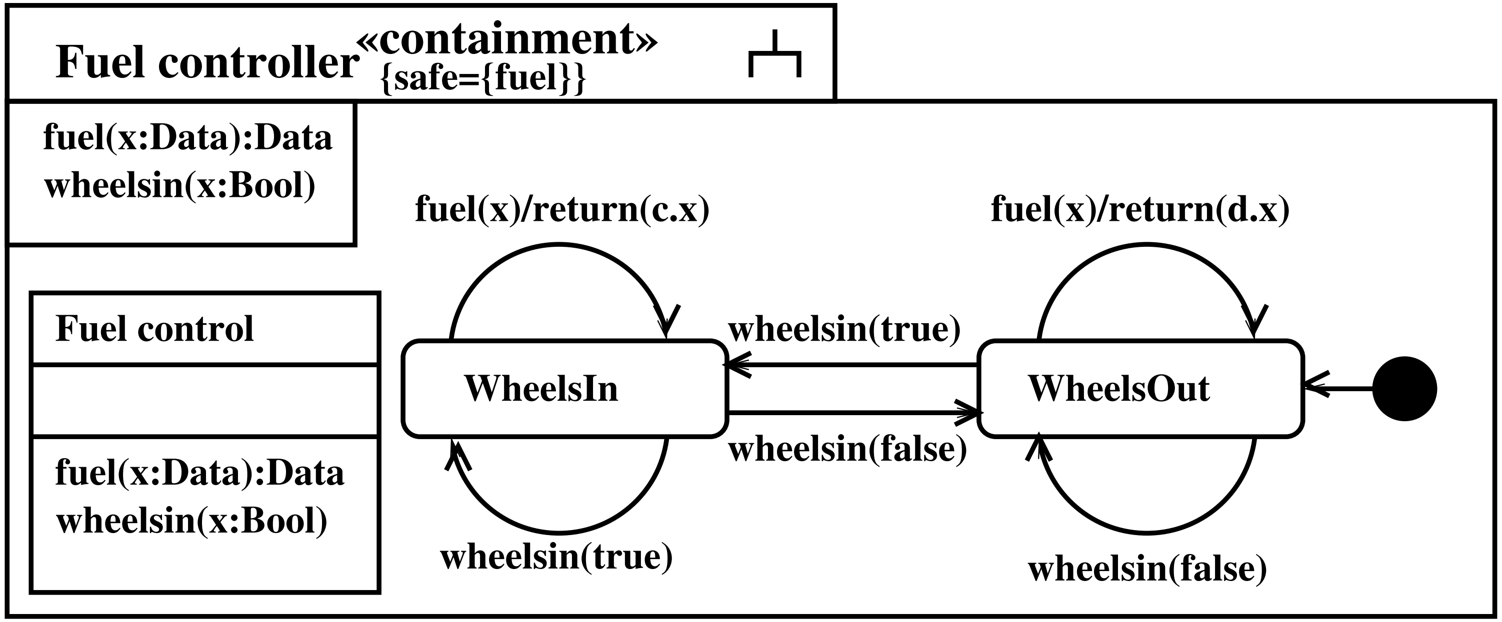
Prevent indirect corruption of data.

Constraint:

Value of any data element *d* may only be influenced by data whose requirements attached to ¿criticalÀ imply those of *d*.

Make precise by referring to execution semantics (view of history associated with safety level).

Example ¿containmentÀ



Violates containment because a {safe} value depends on un{safe} value.

Can check this mechanically.

Other checks

Have other consistency checks such as

- Is the software‘s response to out-of-range values specified for every input ?
- If input arrives when it shouldn't, is a response specified ?

and other safety checks from the literature.

Similarly: UMLsec

Safety = „Security against stupid adversaries“

Security = „Safety for paranoids“

Adversaries in security correspond to failures in safety.

Replace failure model in UMLsafe by adversary model to get UMLsec.

Applications

- Common Electronic Purse Specifications
- Analysis of multi-layer security protocol for web application of major German bank
- Analysis of SAP access control configurations for major German bank
- Risk analysis of critical business processes (for Basel II / KontraG)
- …

Tool-support

Commercial UML modelling tools: so far mainly syntactic checks and some code-generation.

Goal: more sophisticated analysis; connection to analysis tools.

Several possibilities:

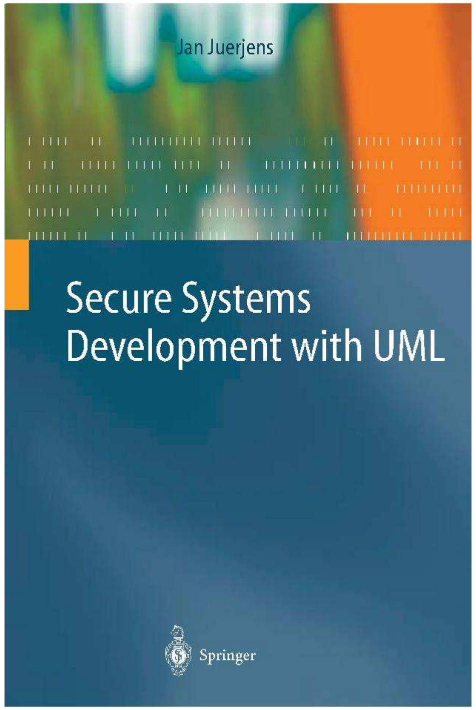
- General purpose language with integrated XML parser (Perl, …)
- Special purpose XML parsing language (XSLT, …)
- Data Binding (Castor; XMI: e.g. MDR)

Connection with AutoFocus

- CASE tool, UML-like notation, formal basis (FOCUS)
- Graphical, view oriented modelling
  - System Structure Diagrams
  - State Transition Diagrams
  - Message Sequence Charts
  - Data Type Definitions
- Features:
  - Simulation
  - Validation (Consistency, Testing, Model Checking)
  - Code Generation (e.g. Java, C, Ada)
  - Connection to Matlab

Some resources

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



Tutorials @ CSS'03, Mexico, 19-21 May; FME'03, Pisa, Sept. ....

More information:

http://www.jurjens.de/jan

Finally

We are always interested in industrial challenges for our tools, methods, and ideas to solve practical problems.

More info: http://www.jurjens.de/jan

Contact me here or via Internet.

Thanks for your attention !