

Critical Systems Development with UML

Jan Jürjens

Software & Systems Engineering
Informatics, Munich University of Technology
Germany



juerjens@in.tum.de

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



Critical Systems Development

High quality development of critical systems (dependable, security-critical, real-time,...) is **difficult**.

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

Quality vs. cost

Systems on which human life and commercial assets depend need **careful** development.

Systems operating under the possibility of failure or attack need to be free from **weaknesses**.

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).

Challenges

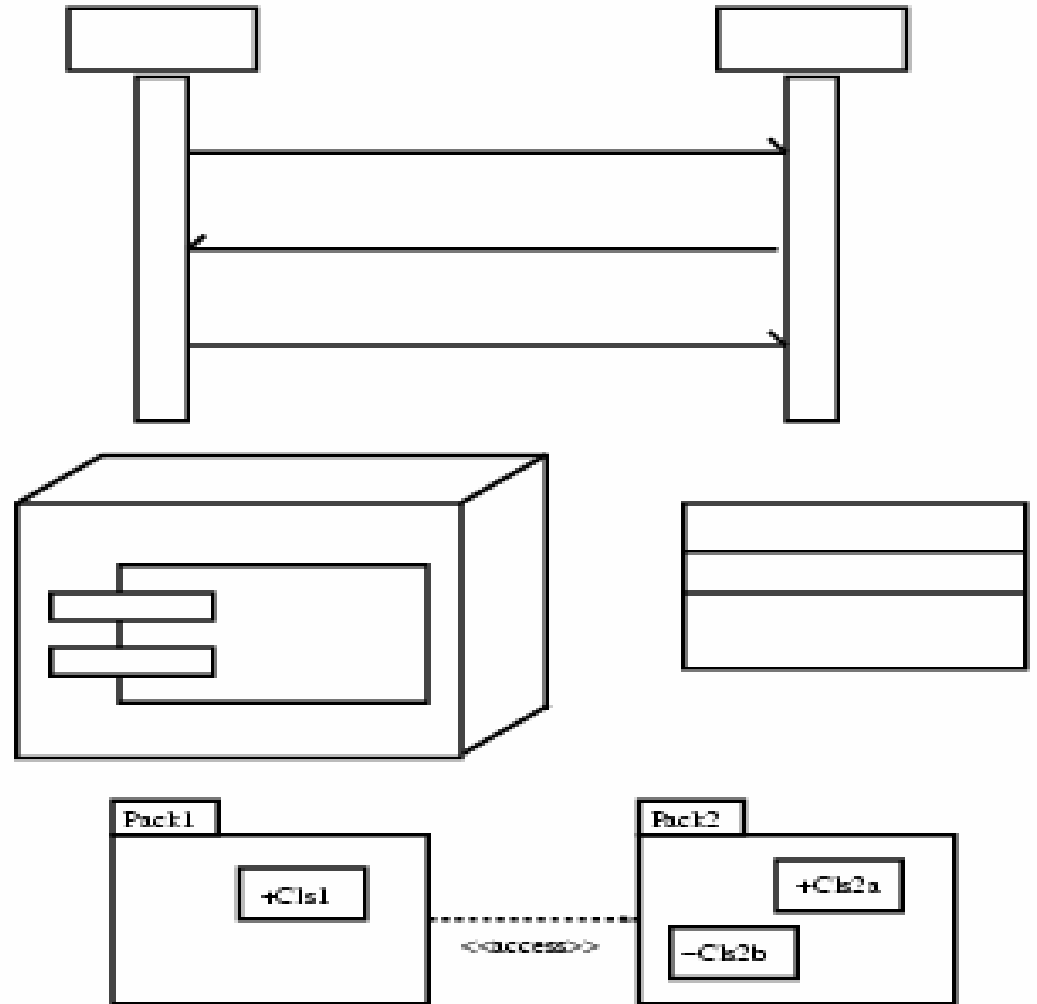
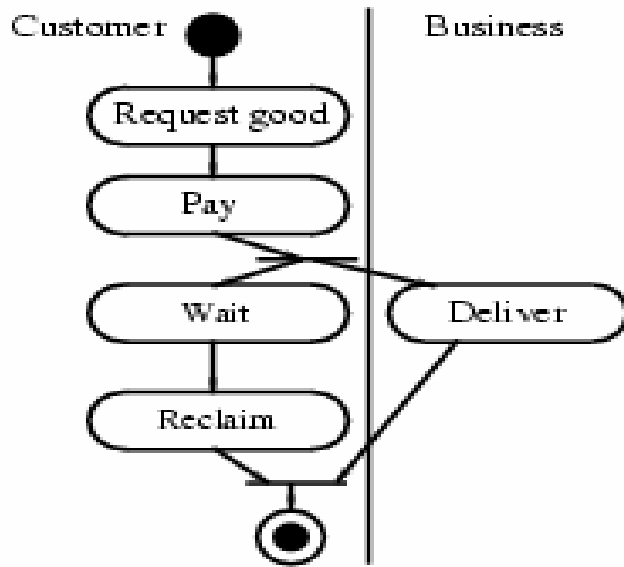
- **Adapt** UML to critical system application domains.
- **Correct use** of UML in the application domains.
- Conflict between **flexibility** and **unambiguity** in the meaning of a notation.
- Improving **tool-support** for critical systems development with UML.

Using UML

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 `<<label>>`.

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

Constraint: **refine** semantics of stereotyped element.

Profile: **gather** above information.

UMLsafe

UMLsafe: extension for dependable **systems** development.

Goals :

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

The UMLsafe profile

Recurring dependability requirements offered as stereotypes with tags.

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

Ensures that stated dependability requirements **provide** desired level of dependability.

Ensures that UML specification **provides** requirements.

Requirements on UML dependability extension

- Provide basic **dependability requirements**.
- Allow considering different **failure scenarios**.
- Allow including important **dependability concepts** (e.g. *redundancy*).

UMLsafe profile (excerpt)

Stereotype	Elements	Tags	Constraints	Description
risk	link, component	delay, corruption, loss		risk to exchanged data
guarantee	link, component	maxdelay, avdelay, delivery		required guarantees on data
safe links	subsystem		dependency dependab. matched by links	physical layer gives dependab.
safe	subsystem		dependencies respect dependability	communication pres. dependab.
dependency containment	subsystem		data only influenced by data with same depend.	data exchange preserves dependab.

Failures

Data from other components may be

- delayed
- corrupted
- lost.

Often, failures occur **randomly** (e.g. hardware).

Redundancy model determines which level of redundancy provided.

Failure models

For redundancy model R , stereotype s , have set $\text{Failures}_R(s) \subseteq \{\text{delay}_d, \text{corrupt}_c, \text{lose}_l\}$ where

- d : distribution on $R \geq 0$ (time delay of messages),
- c : $\text{Exp} \rightarrow \text{Exp}$: probabilistic function (gives corrupted version of message),
- l is a distribution on $\{0,1\}$ (0 represents loss)

(in each case **incorporating** redundancy).

Example

Suppose redundancy model R uses controller with redundancy 3 and the fastest result.

Then

- $d(r)$: probability that the fastest controller takes time r ($r \in R \geq 0$),
- c : corruption function for single controller,
- $l(0) = p^3$ where p probability that single controller fails.

<<risk>>

Describe risks connected with use of communication **links** resp. system **nodes**.

Tags: {delay=d}, {corruption=c}, {loss=l} where

- d : distribution on $R \geq 0$
- c : $\text{Exp} \rightarrow \text{Exp}$: probabilistic function,
- l is a distribution on $\{0, 1\}$

Tags on nodes concern connected links.

Can define special stereotypes for recurring risk scenarios.

<<guarantee>>

Describe guarantees required from communication **dependencies** resp. system **components**.

Tags: {maxdelay= m }, {avdelay= a },
{delivery= d } where

- m, a : time spans,
- d : probability.

<<safe links>>

Ensures that dependability requirements on **communication** and **components** met by physical layer.

Constraint: there exists no dependency d with stereotype <<guarantee>> between components on nodes n, m , with a communication link l between n and m with stereotype <<risk>> such that the guarantees in the tags associated with <<guarantee>> cannot be enforced given the risks in the tags associated with <<risk>>.

A

A

<<critical>>

Mark objects with critical data.

Given a lattice SL of safety levels, have associated tags $\{s/=Data\}$ where $s/\in SL$ and $Data$ is a set of data values.

<<safe dependency>>

Ensures that communication dependencies between components **respect** dependability requirements on communicated data given by <<critical>>.

Assume functions *md*, *ad*, *del* from safety levels to the maximum delay, average delay, and delivery probability bounds permitted by the safety level.

Constraint: given dependency *d* stereotyped <<guarantee>> from *C* to *D*:

- The requirements on data in *C* are implied by those in *D* on the same data.
- The requirements on data in *C* also appearing in *D* are met by the guarantees of *d*.

<<containment>>

Prevent indirect corruption of data.

Constraint:

Value of any data element *d* may only be influenced by data whose requirements attached to <<critical>> imply those of *d*.

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).

Similarly: UML_{sec}

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**.

Connection with analysis tool

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

Work in progress: connect to **verification** tools.

Industrial CASE tool with UML-like notation:

AUTOFOCUS

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

- verification
- code generation
- test-sequence generation

Resources

Book on related UMLsec:

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 !