A Framework for Simulation Models of Service-Oriented Architectures

Falko Bause, Peter Buchholz, Jan Kriege, Sebastian Vastag

CRC 559, Computer Science IV

June 28, 2008

Outline

Motivation

Service-Oriented Architecture (SOA)

represents a collection of best practices principles and patterns related to service-aware, enterprise-level, distributed computing. *(Def. from OASIS)*

- QoS, SLAs, ...
  $\Longrightarrow$ performance issues are of interest.
- abstract performance models, e.g. manual derivation of analytical models (QNs)
- simulation models
  - cumbersome, error-prone
  - should consider both:
    "enterprise-level" + communication
    (different notations: BPEL, process chains vs. simulation language for networks, protocols)

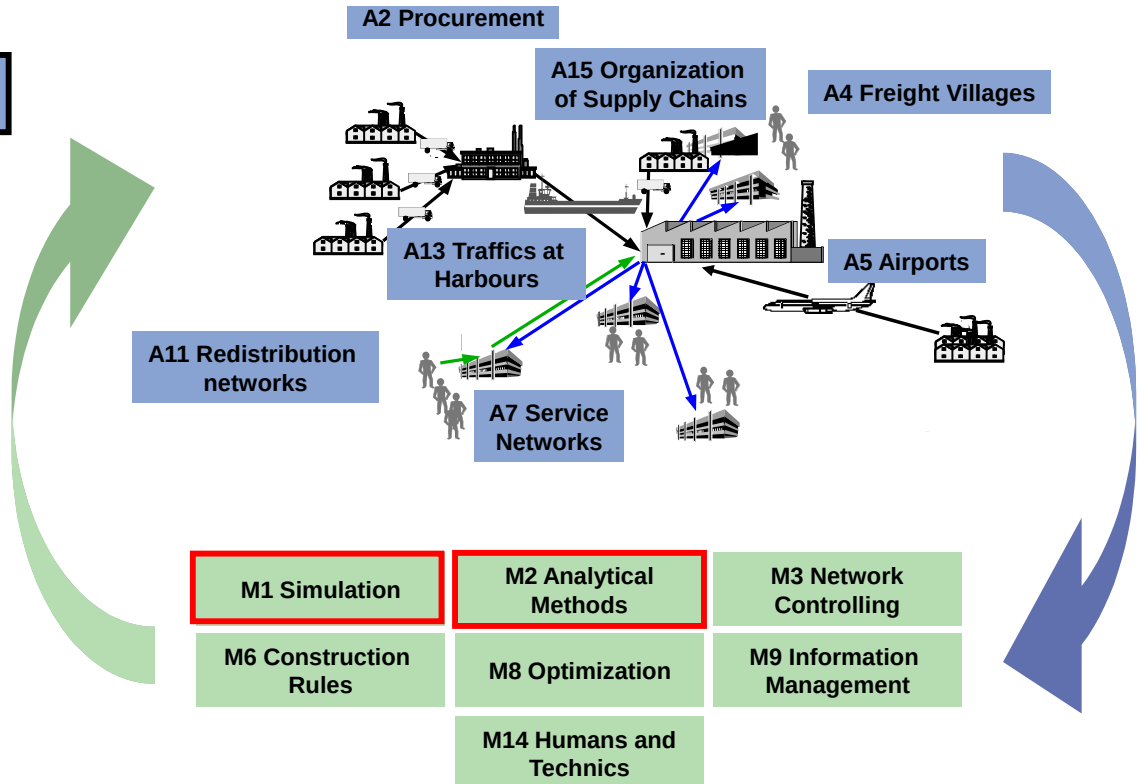
Our approach

Hybrid Specification

- process chains for "enterprise-level"
- *OMNeT++* for network

Collaborative Research Center 559 (1998-2008): Modelling of Large Logistics Networks

(mechanical engineering, economics, computer science, statistics)

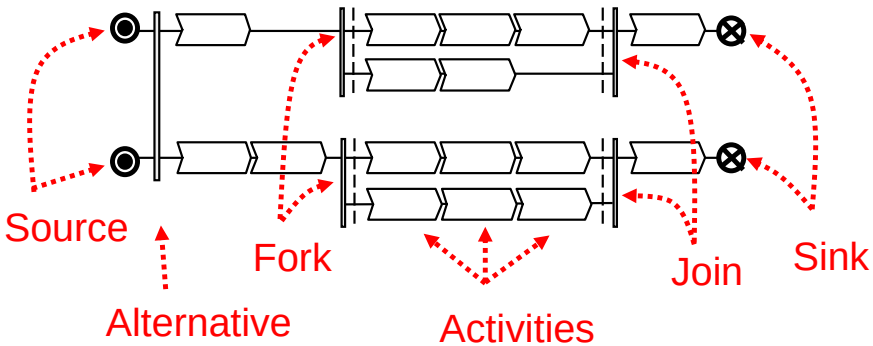


≈ 60 scientists in 14 sub-projects

Collaborative Research Center 559 (1998-2008): Modelling of Large Logistics Networks

"Common Language": Process Chain Paradigm (by Kuhn)

Process Chains describe process patterns



1998 descriptive, "semi-formal" (manual transformation into simulation models (additional effort, inconsistencies))

98-08 development of *ProC/B* $\subset$ Process Chains (for automated transformation into simulation models)

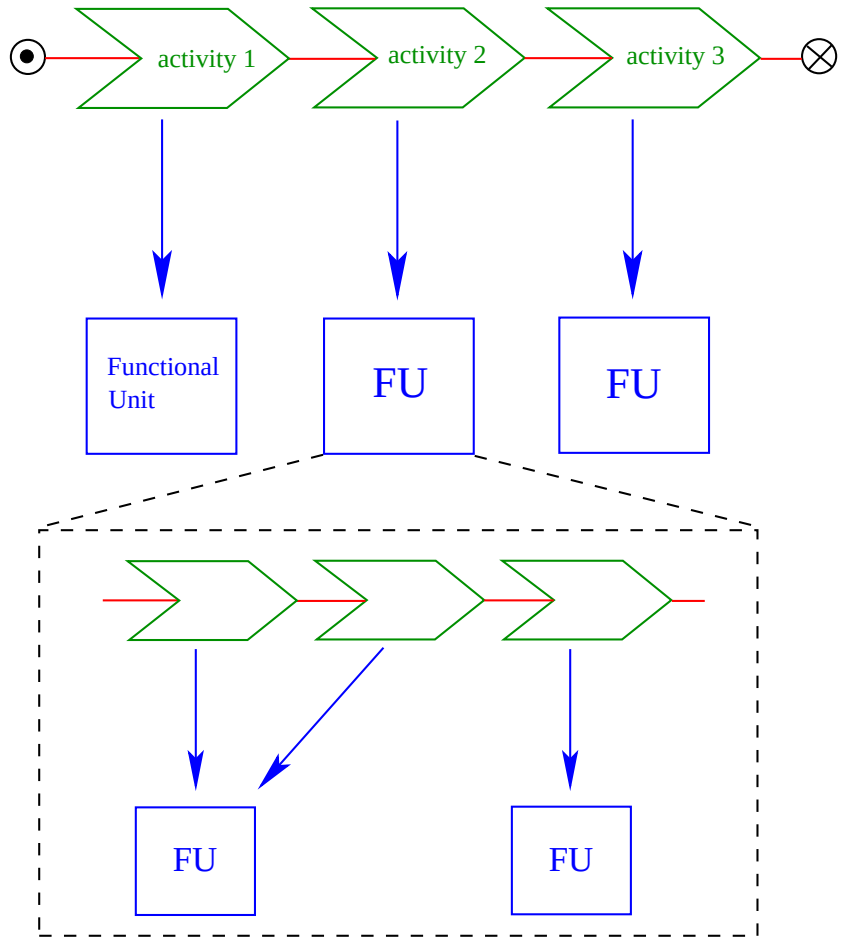
Process Chains, general idea

Process chains describe behaviour:
What happens and when?

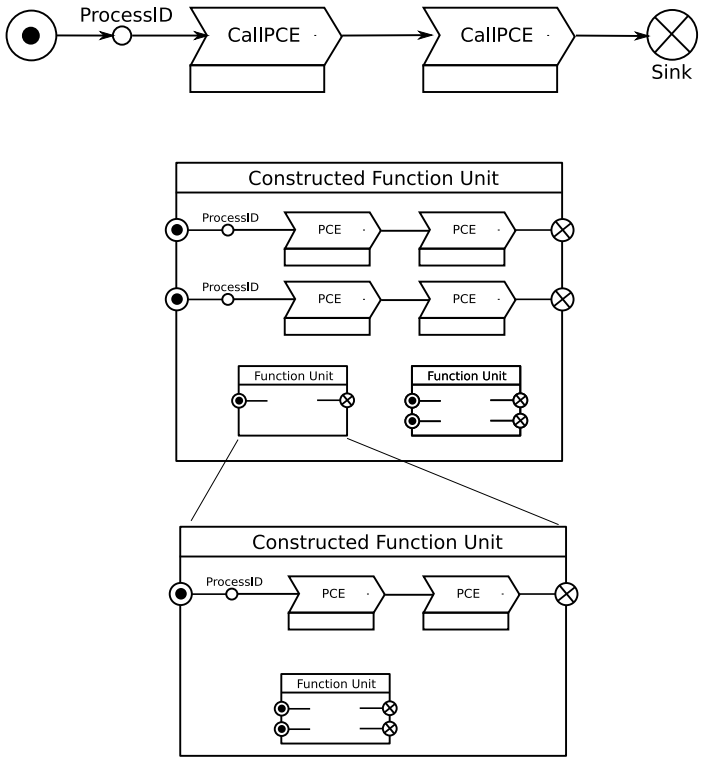
- activities: PC elements
- sequencing by: sequential concatenation + connectors
- process incarnation + termination: **sources + sinks**

ProC/B, general idea

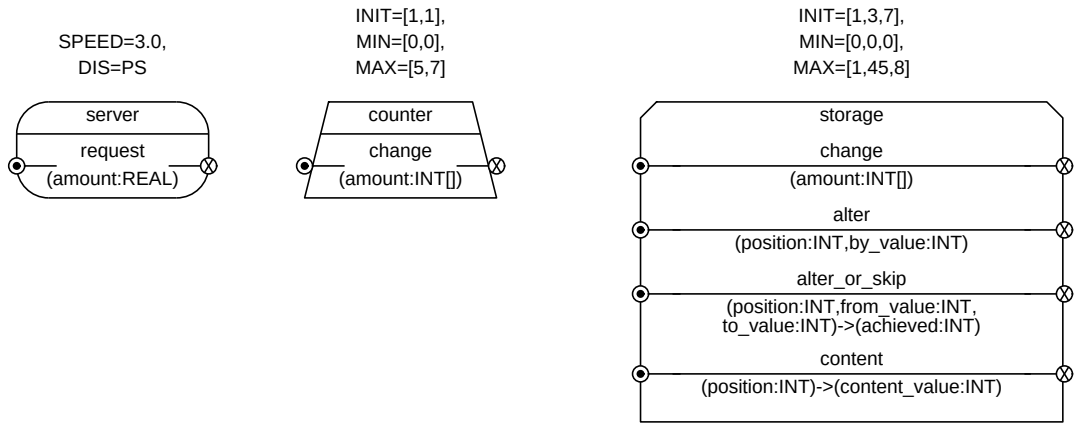
- **Who performs activities?**
  - **"resources" /functional units capture system structure**
- **How is it done?**



ProC/B hierarchies

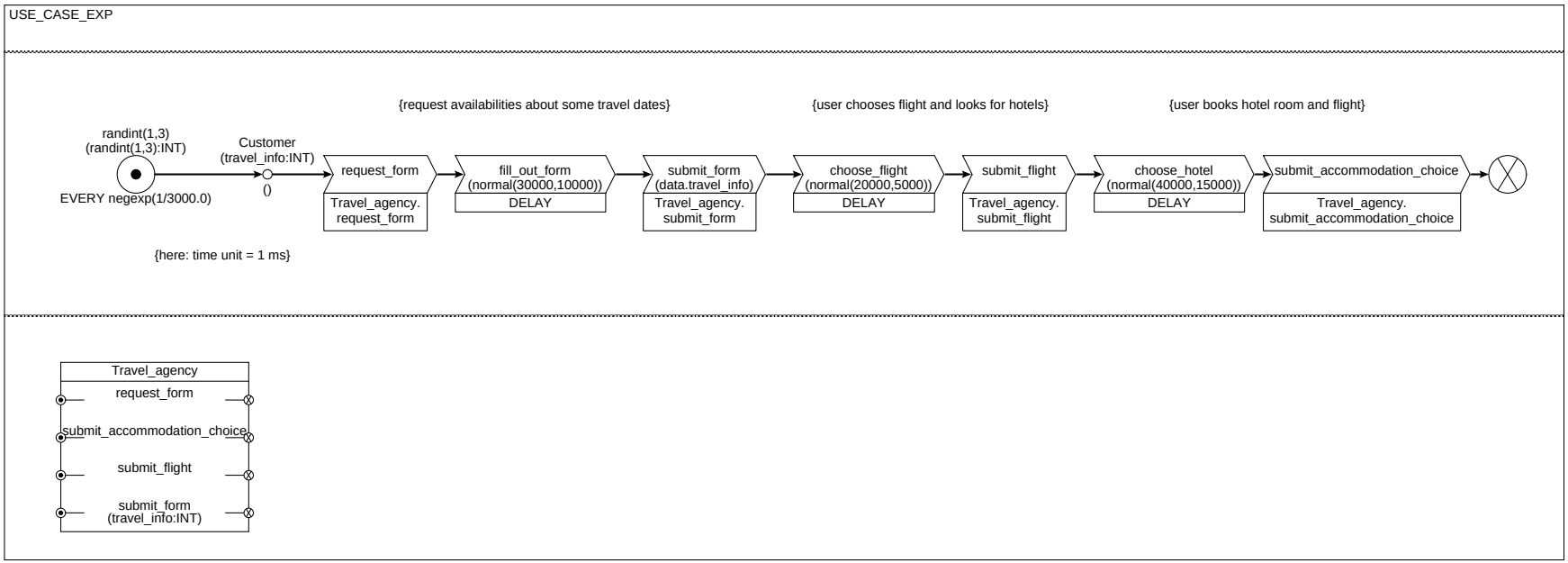


Hierarchie ends at standard (pre-defined) FUs:

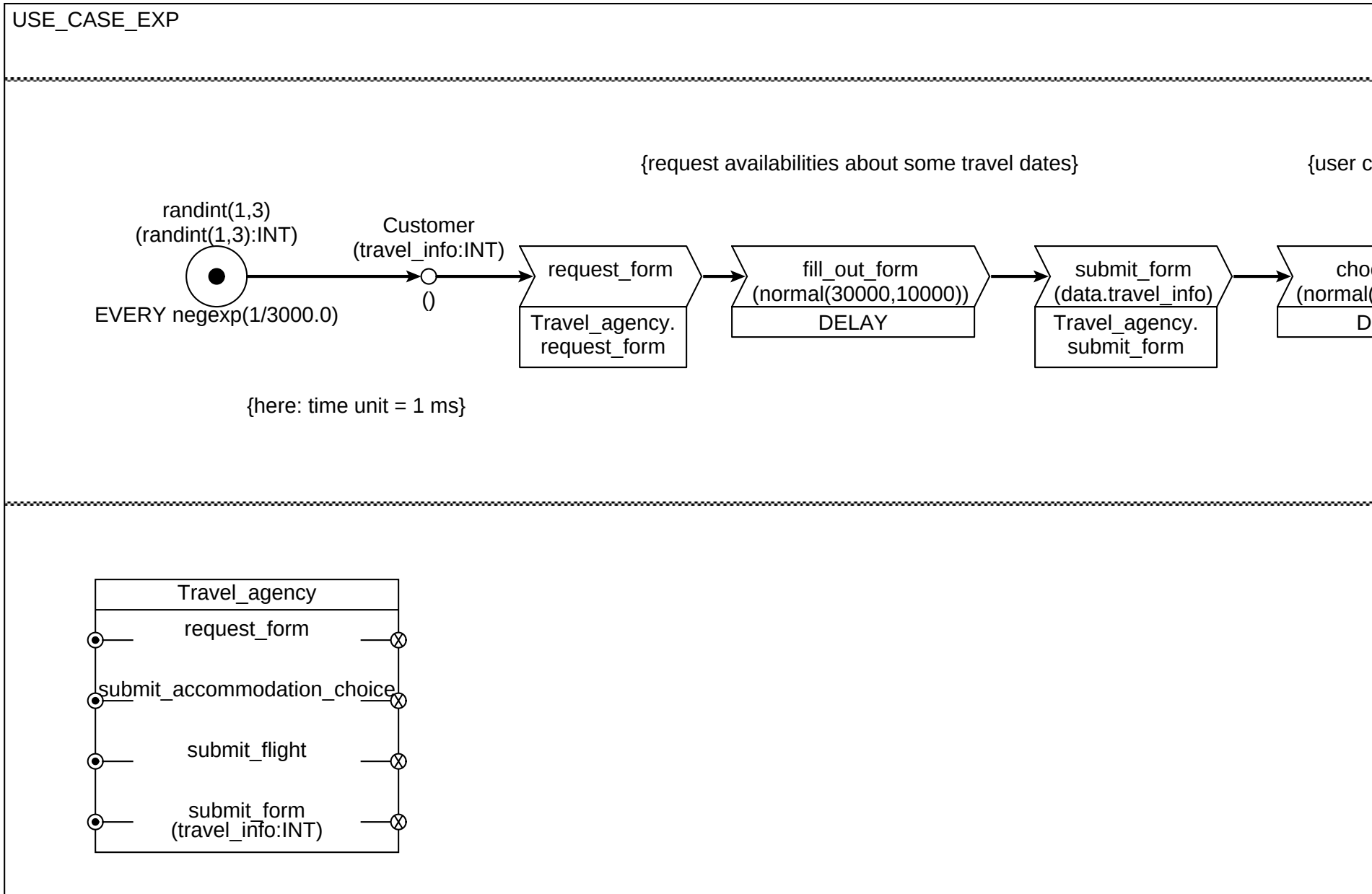


- servers "modelling time" (similar to queues of QNs)
- counters "modelling space" (act like (multi-dimensional) semaphors) storages/buffers: "user-friendly" semaphors

ProC/B example

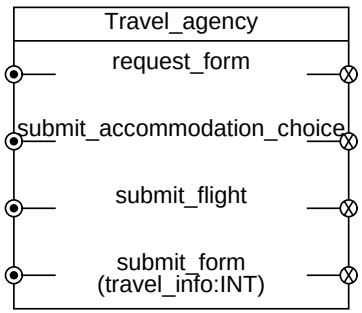


ProC/B example

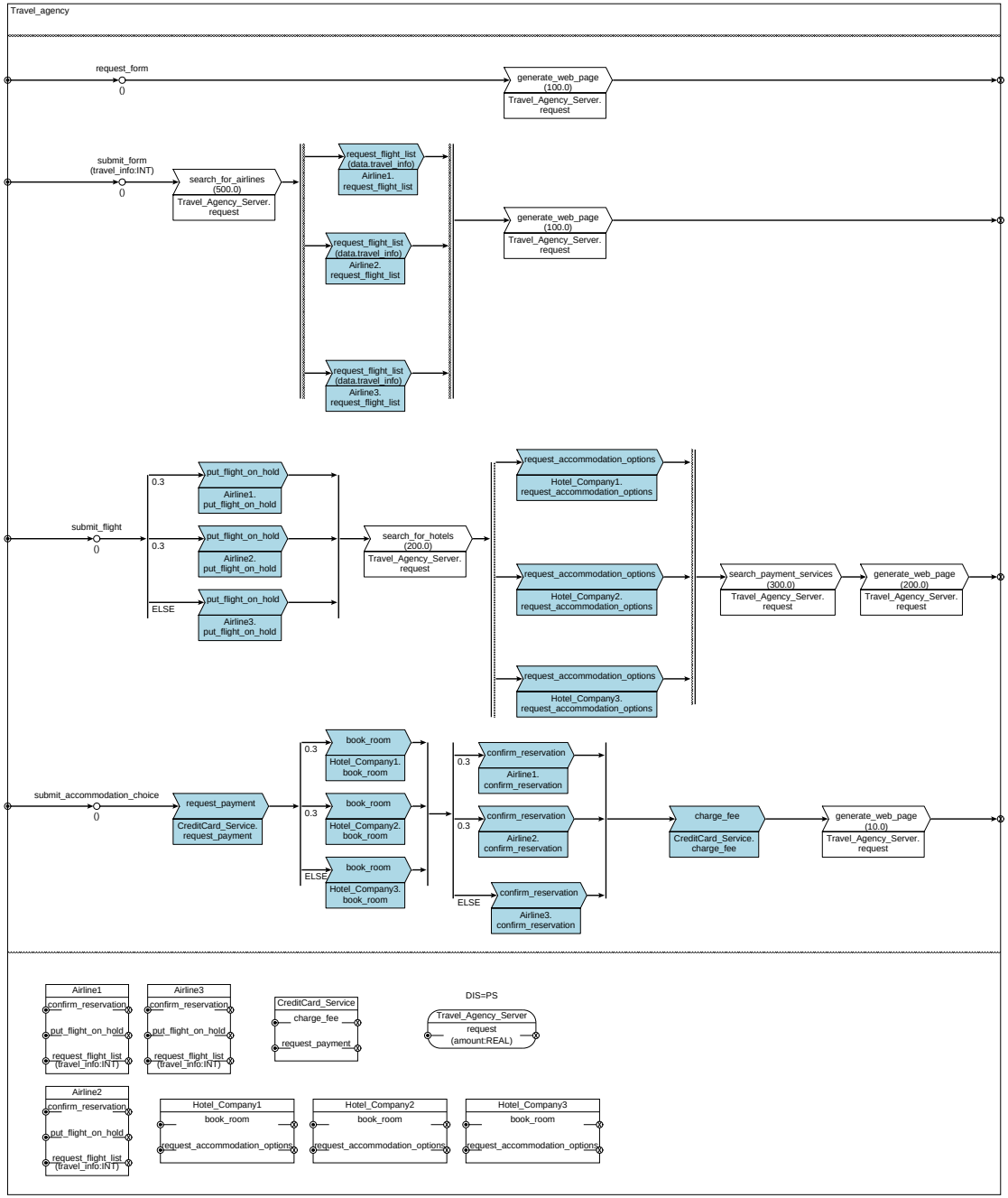


ProC/B example (cont'd)

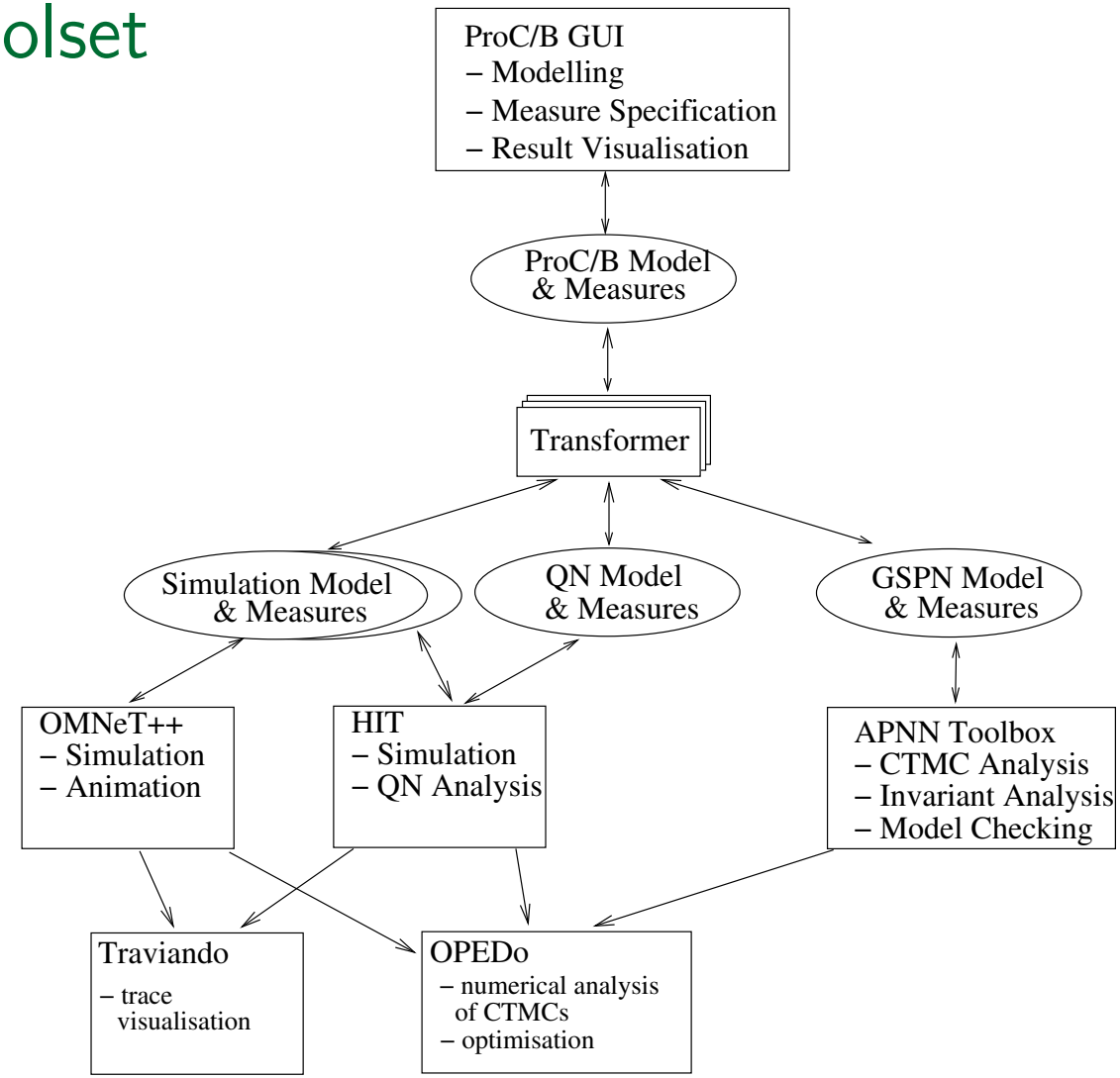
external view of FU



internal view of FU



ProC/B Toolset



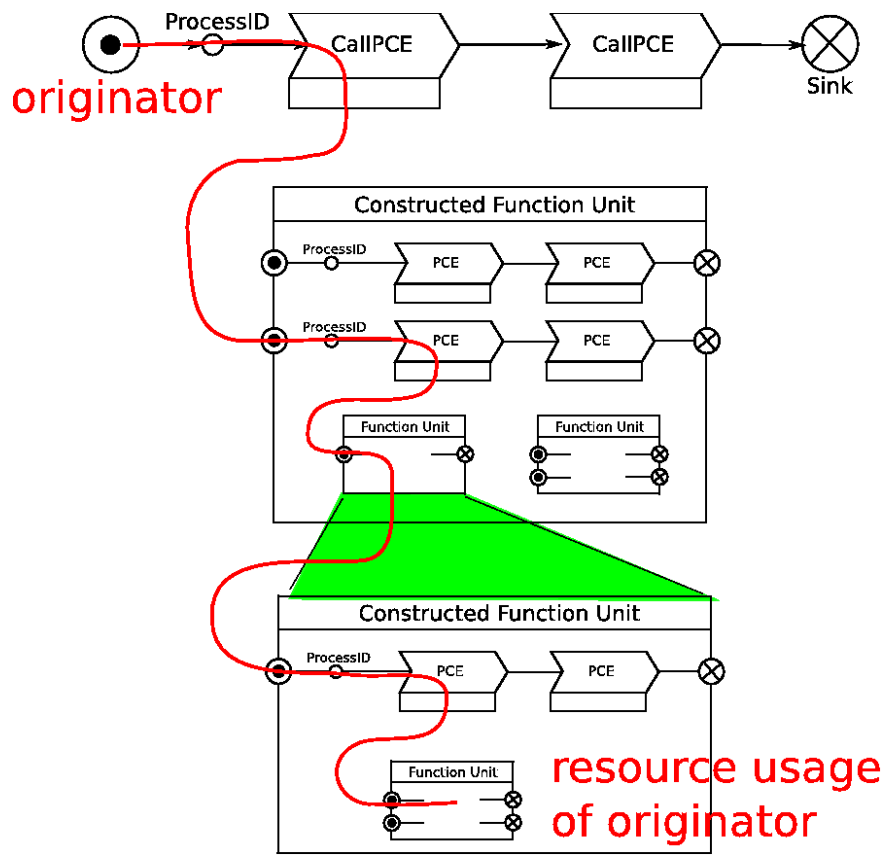
ProC/B Results

- "standard" performance measures: throughput, population, response time, ...
- user defined measures ("REWARDS")

All measures can be discriminated with respect to the originator,

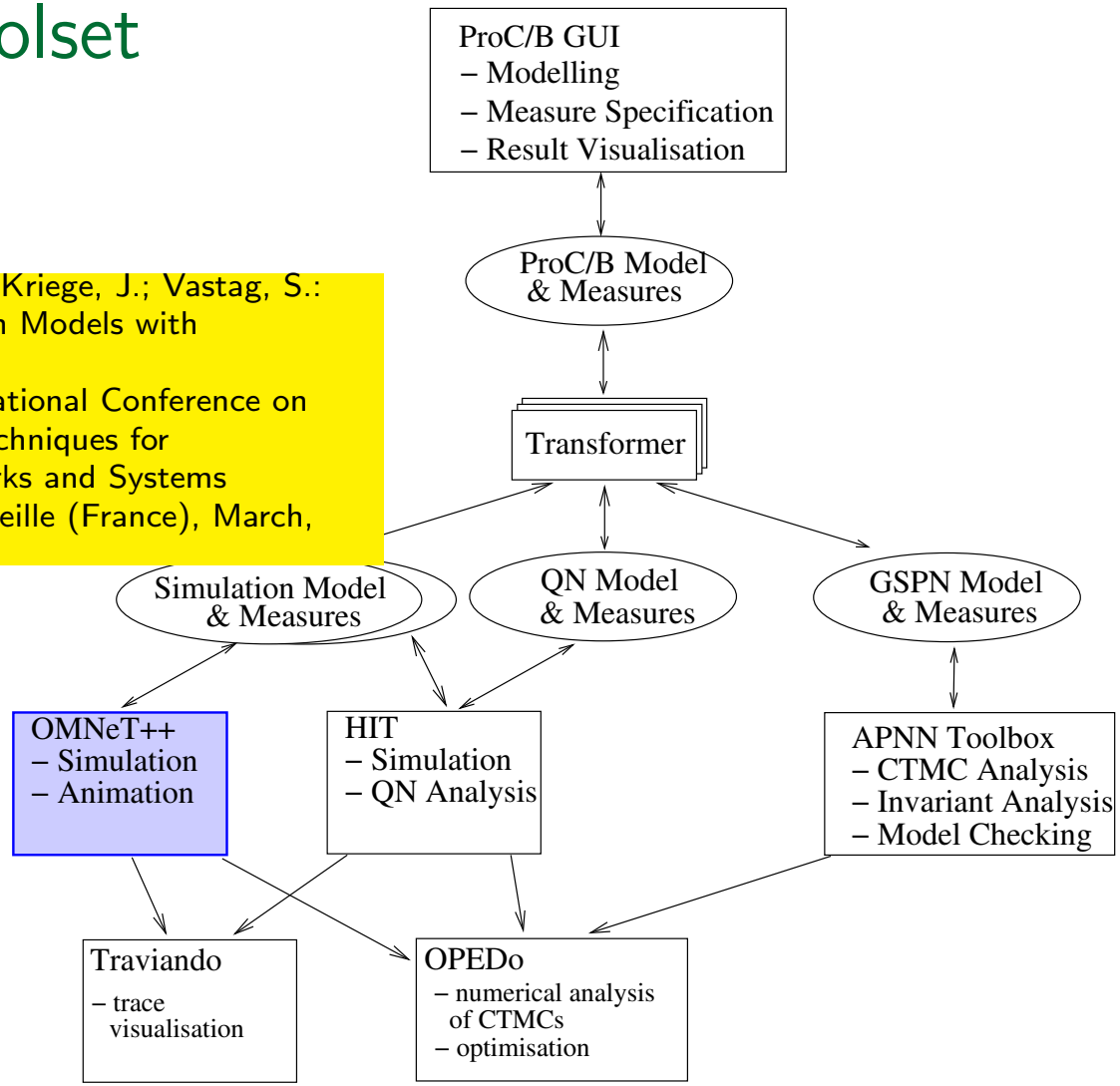
e.g. throughput due to ...

(supports "cost accounting")



ProC/B Toolset

Bause, F.; Buchholz, P.; Kriege, J.; Vastag, S.: Simulating Process Chain Models with OMNeT++. Proc. of the First International Conference on Simulation Tools and Techniques for Communications, Networks and Systems (SIMUTools 2008), Marseille (France), March, 2008.



OMNeT++

see http://www.omnetpp.org/

"is a public-source, component-based, modular and open-architecture simulation environment with strong GUI support and an embeddable simulation kernel. Its primary application area is the simulation of communication networks . . ."

- simple modules (behaviour implemented as C++ classes)
- compound modules (support hierarchical modelling)
- messages and channels (modules communicate via messages)

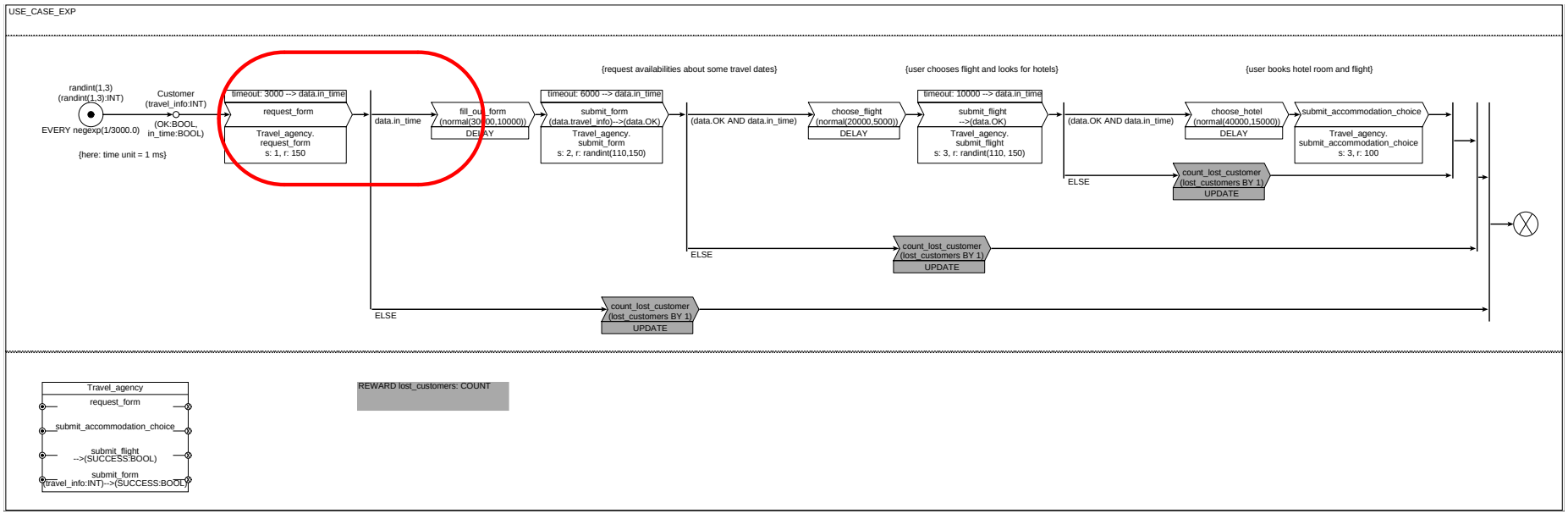
Frameworks are available, one is the INET Framework, which contains IPv4, IPv6, TCP, UDP protocol implementations, and several application models.

Support for SOA scenarios

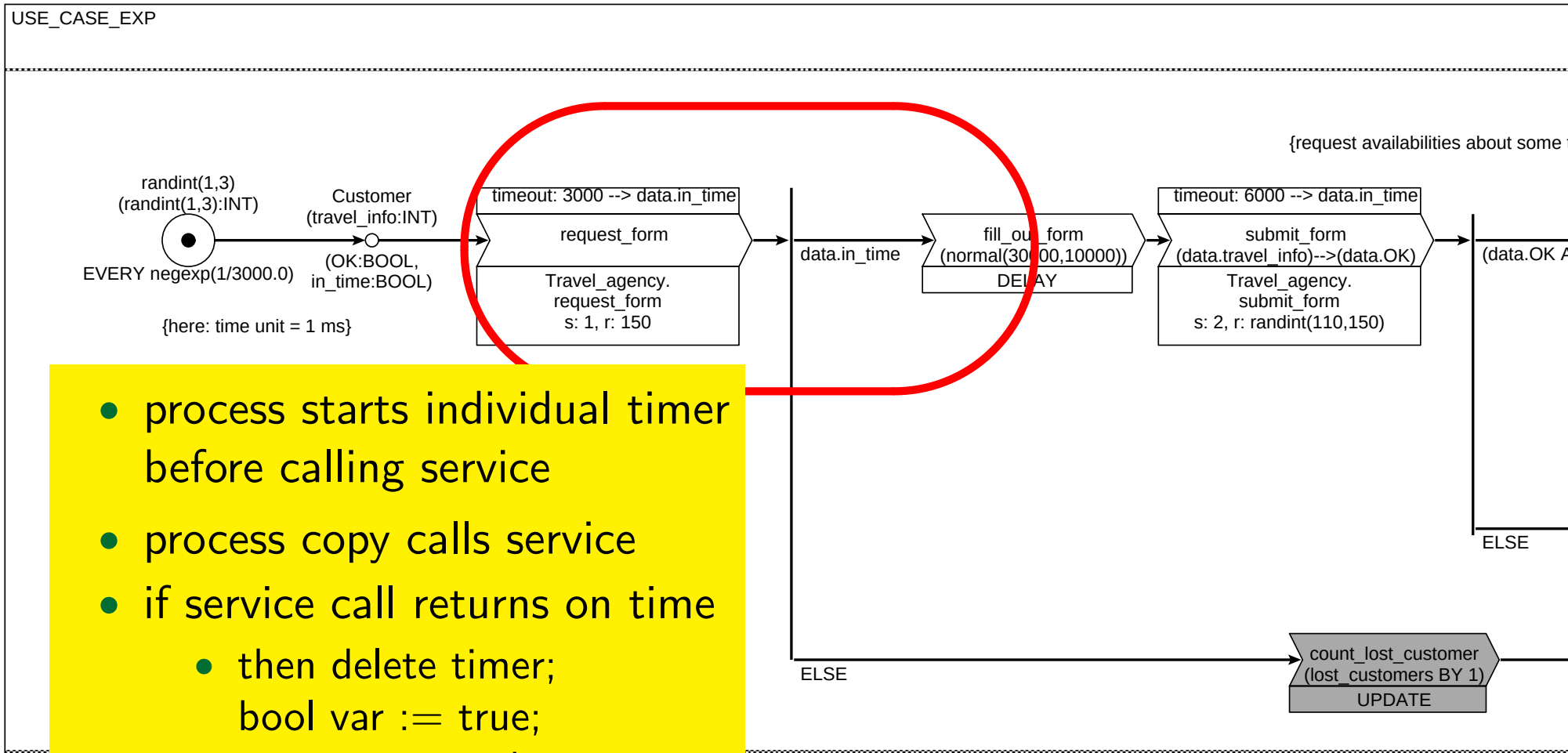
- Integration of a timeout mechanism into *ProC/B*, for modelling user behaviour

- Combining *ProC/B* and INET models (into a single *OMNeT++* model), for modelling network delays

ProC/B example with timeouts

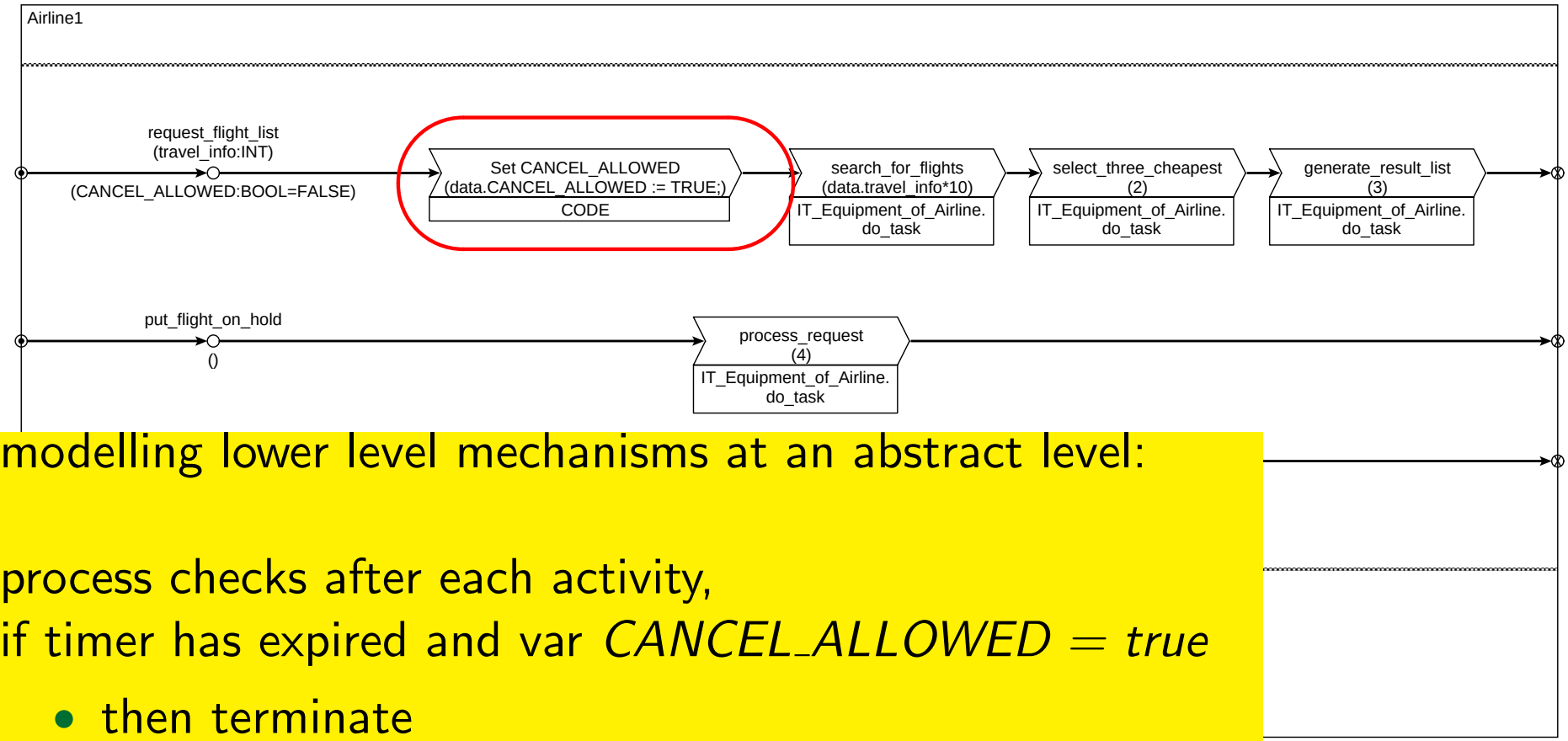


ProC/B example with timeouts



- process starts individual timer before calling service
- process copy calls service
- if service call returns on time
  - then delete timer;
    bool var := true;
    process proceeds
  - else bool var := false;
    process proceeds (default return values);
    late return of service call will be ignored

ProC/B example with timeouts (cont'd)



modelling lower level mechanisms at an abstract level:

process checks after each activity,
if timer has expired and var *CANCEL_ALLOWED* = *true*

- then terminate
- else proceed

Support for SOA scenarios

- Integration of a timeout mechanism into *ProC/B*, for modelling user behaviour

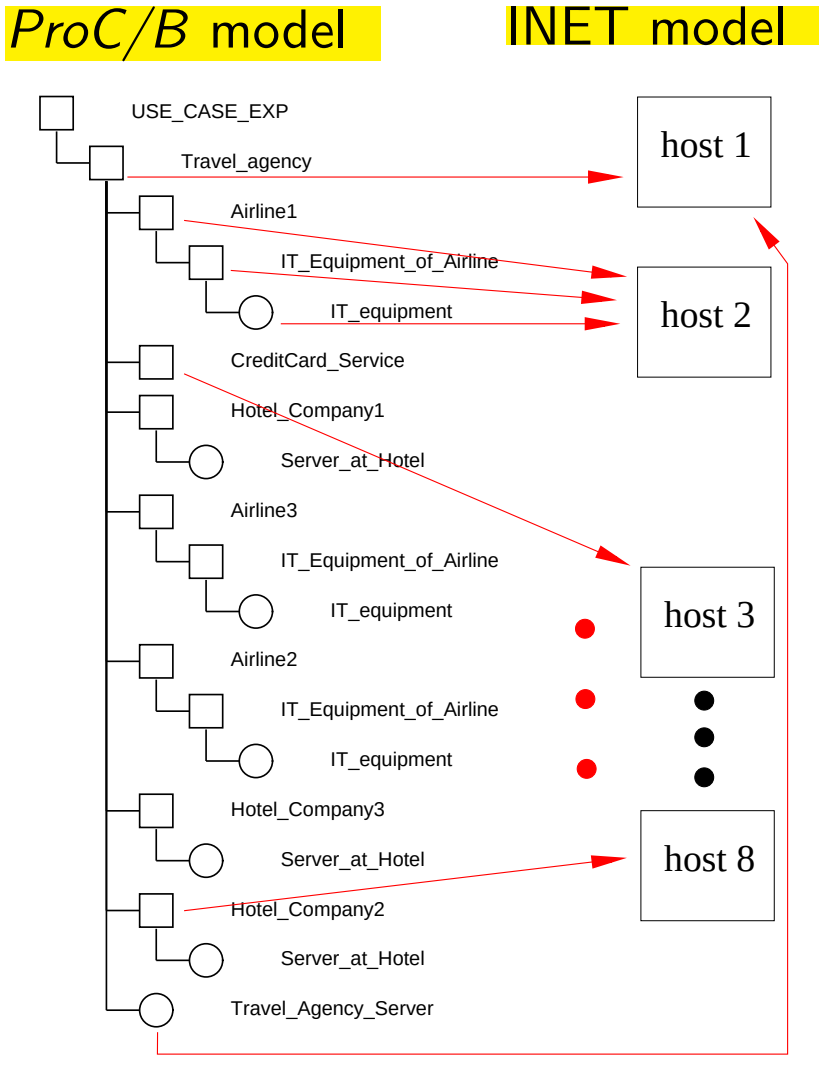
- Combining *ProC/B* and INET models (into a single *OMNeT++* model), for modelling network delays

Combining *ProC/B* and INET models

Typical INET models specify load generation in host modules.
Main idea:

- all activities of an FU are executed on one host
- define mapping FUs $\mapsto$ hosts
  (need neither be injective nor surj.)

  FU A $\mapsto$ host X
  FU B $\mapsto$ host Y
- process in FU A calls service of FU B;
  this initiates exchange of messages between hosts X and Y (if $X \neq Y$)

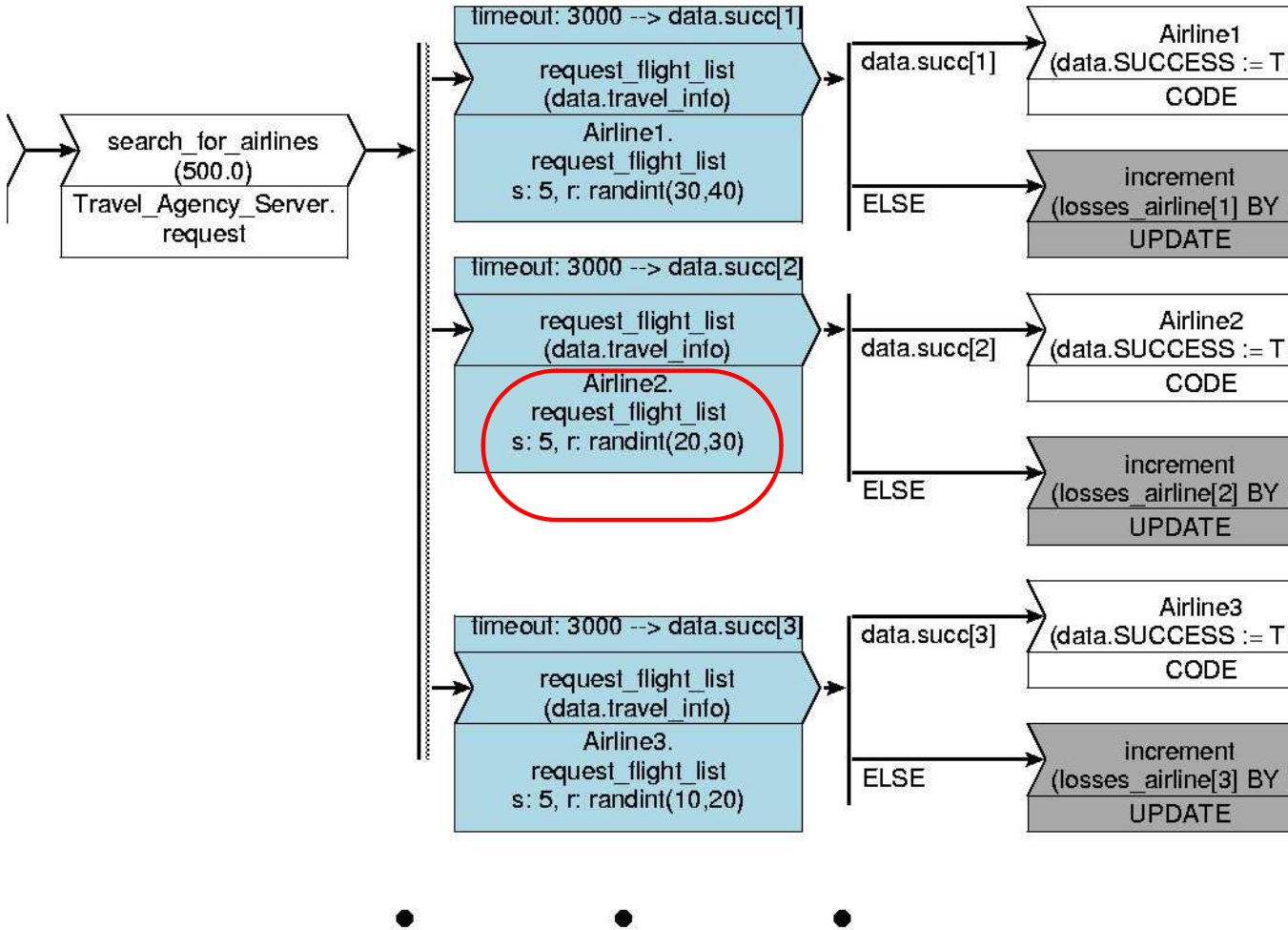


Combining *ProC/B* and INET models (cont'd)

Additional *ProC/B* annotations

- s: amount of data sent
- r: amount of data received

(values interpreted in INET part)



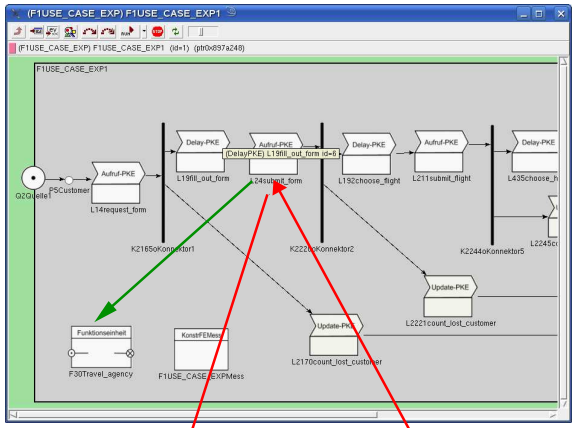
Combining *ProC/B* and INET models (cont'd)

E.g. TCP transmission delay for *ProC/B* remote service call:

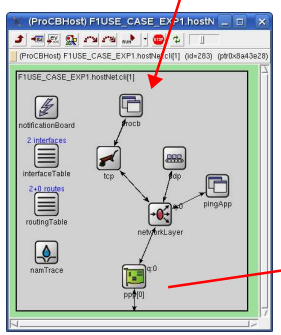
- process is suspended
- "INET message" is sent to (modified) host module
- host initiates TCP connection and transmits specified amount of data
- after peer closes connection *ProC/B* process proceeds, i.e. *ProC/B* service call starts

Advantages:

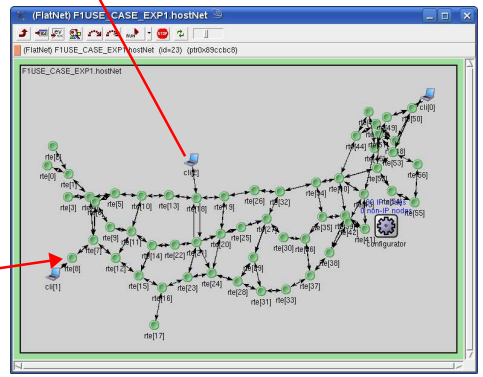
- change of INET model only at hosts
- consistent statistics of *ProC/B* part



OMNeT++-Representation of transformed ProC/B-model



host



INET model

Example results

Lost customers per second

customer mean inter-arrival time (sec.)	network delay between two routers of INET model	0.001s	0.01s	0.05s	0.075s	0.1s
4	lost customers per sec.	0.0650	0.0687	0.1318	0.1879	0.4976
	standard deviation	0.3517	0.3560	0.4982	0.5538	0.7371
	confidence 90%	20.00%	13.21%	16.63%	11,27%	9.76%
	relative loss	13.0%	13.7%	26.4%	37.6%	99.5%
3	lost customers per sec.	0.2007	0.1965	0.2768	0.3754	0.6708
	standard deviation	0.6110	0.6040	0.7003	0.7922	0.8965
	confidence 90%	10.25%	6.30%	10.27%	8.39%	4.47%
	relative loss	30.1%	29.5%	41.5%	56.3%	100%
2	lost customers per sec.	0.5903	0.6245	0.6795	0.7432	0.9856
	standard deviation	1.0100	1.0693	1.1395	1.1632	1.2273
	confidence 90%	10.00%	10.88%	5.19%	10.24%	3.69%
	relative loss	59.0%	62.5%	68.0%	74.3%	98.6%

(customers arrive in batches of sizes 1,2,3; sim length = 10000 sec. model time)

Conclusions

- Support for modelling SOA scenarios:
  - *ProC/B* (process chains) allow for modelling enterprise-level ("...,Web services, orchestration,...", timers for user behaviour)
  - INET framework models lower network activities

  both model parts are
  combined into a single (*OMNeT++*) simulation model

- Future work:
  - automated simulation model generation for INET models with appropriate host definitions
    (currently INET models (especially the hosts) are adjusted manually)
  - user-friendly support for asynchronous communication in *ProC/B*