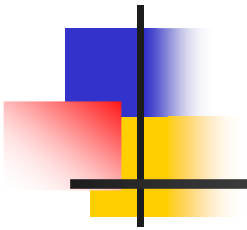
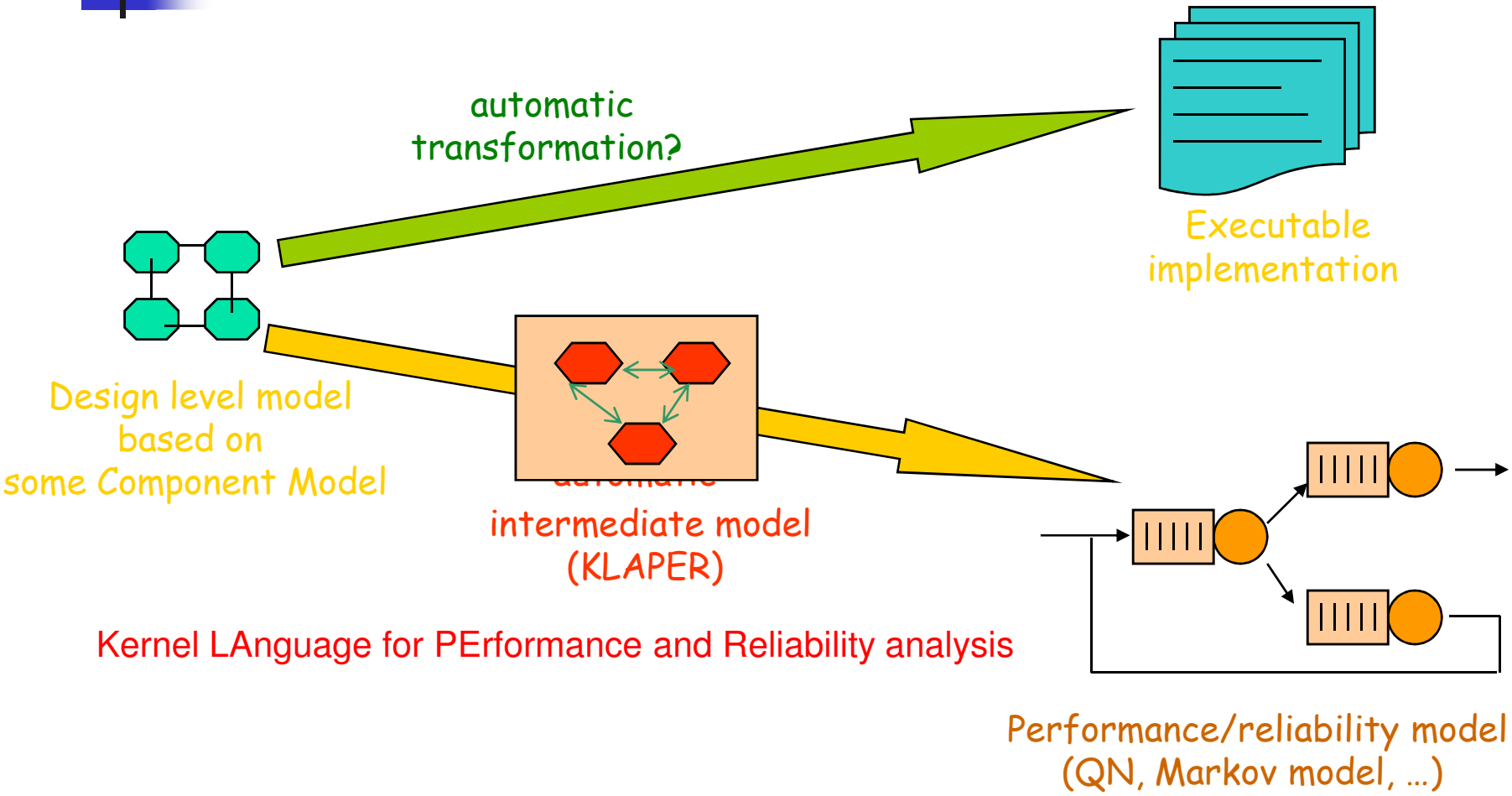


KLAPER: an Intermediate Language for Model-Driven Predictive Analysis of Performance and Reliability

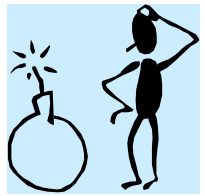
Vincenzo Grassi
Dipartimento di Informatica, Sistemi e Produzione, Università di Roma Tor Vergata
Raffaela Mirandola
Dipartimento di Elettronica e Informazione, Politecnico di Milano
Enrico Randazzo
Dipartimento di Informatica, Sistemi e Produzione, Università di Roma Tor Vergata
Intecs S.p.A., Roma, Italy
Antonino Sabetta
ISTI-CNR, Pisa, Italy

Our contribution



QoS-driven component/connectors selection and composition

Non obvious correlation between the QoS of the assembled architecture and the individual component/connector QoS



assembly QoS *monitoring* to assess the fulfillment of some QoS goal, **after** the components/connectors selection and composition

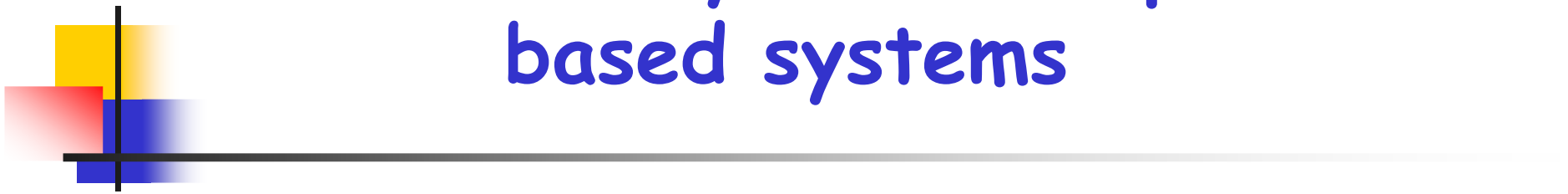
assembly QoS *prediction* **to drive** the selection of components/connectors

need of QoS prediction methodologies

compositional (to exploit the architecture structure)

automatic (to make predictive analysis really effective)

Predictive analysis of component-based systems

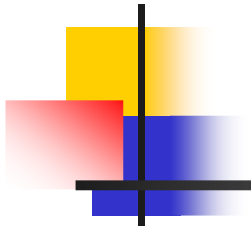


Useful *to drive* the selection and composition of components
- late problem fixing may be too costly

Focus on *extra-functional* attributes
- *performance*
  - throughput
  - completion time
- *reliability*

Predictive analysis must be carried out on ***models*** of the system!

Predictive analysis of component-based systems: main issues



analysis-oriented model building is a complex activity:

- Extraction of relevant Performance/Reliability oriented information from a design oriented model
  - which information?
  - how is it represented?

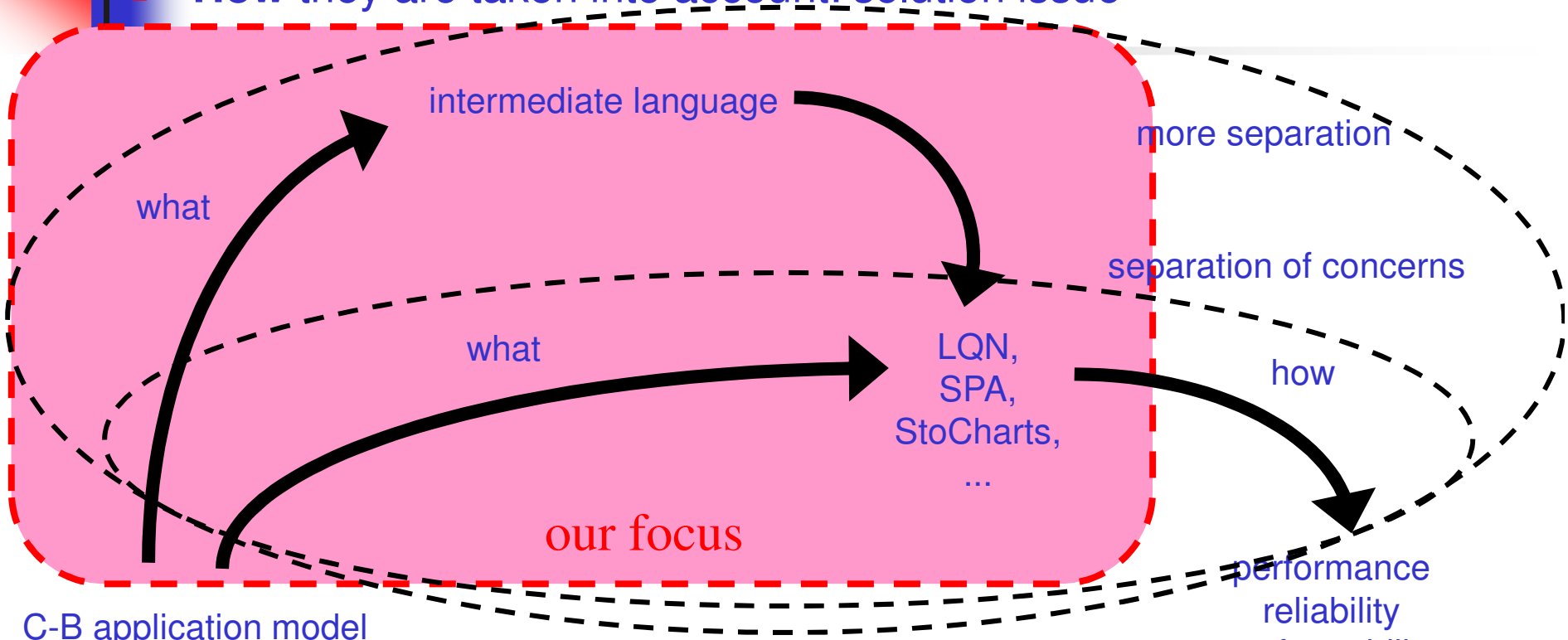
- Construction of performance/reliability/performability analysis-oriented models
  - MDD based approach

- Model solution (not an issue here)

generating a performance/reliability model: the "what" & the "how"

What factors affect the extra-functional attribute: description issue

How they are taken into account: solution issue



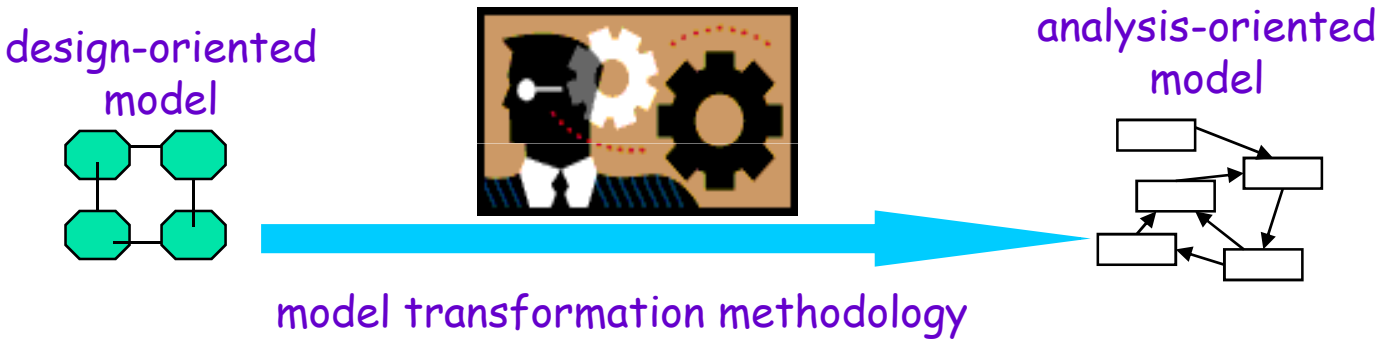
C-B application model
+
"context" information

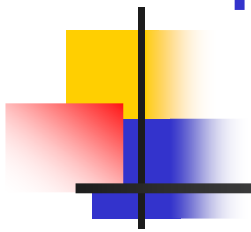
"all-in-one" approach (what+how)

considering factors such as:
contention,
layered architecture,

(context = usage profile, competing appls.,
platform (.NET, J2EE, …), OS, Hw, ...

So far...

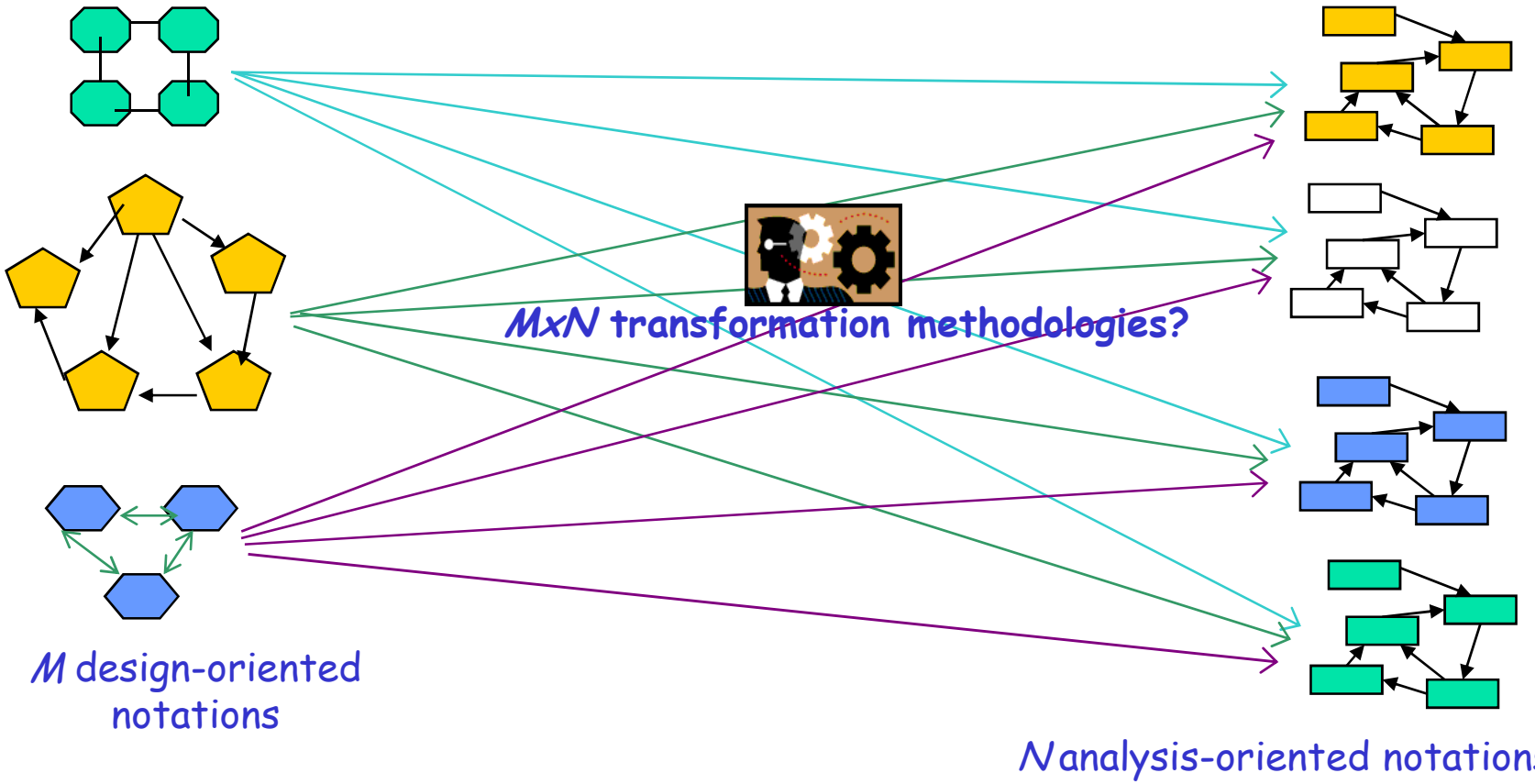

design-oriented model
analysis-oriented model
model transformation methodology



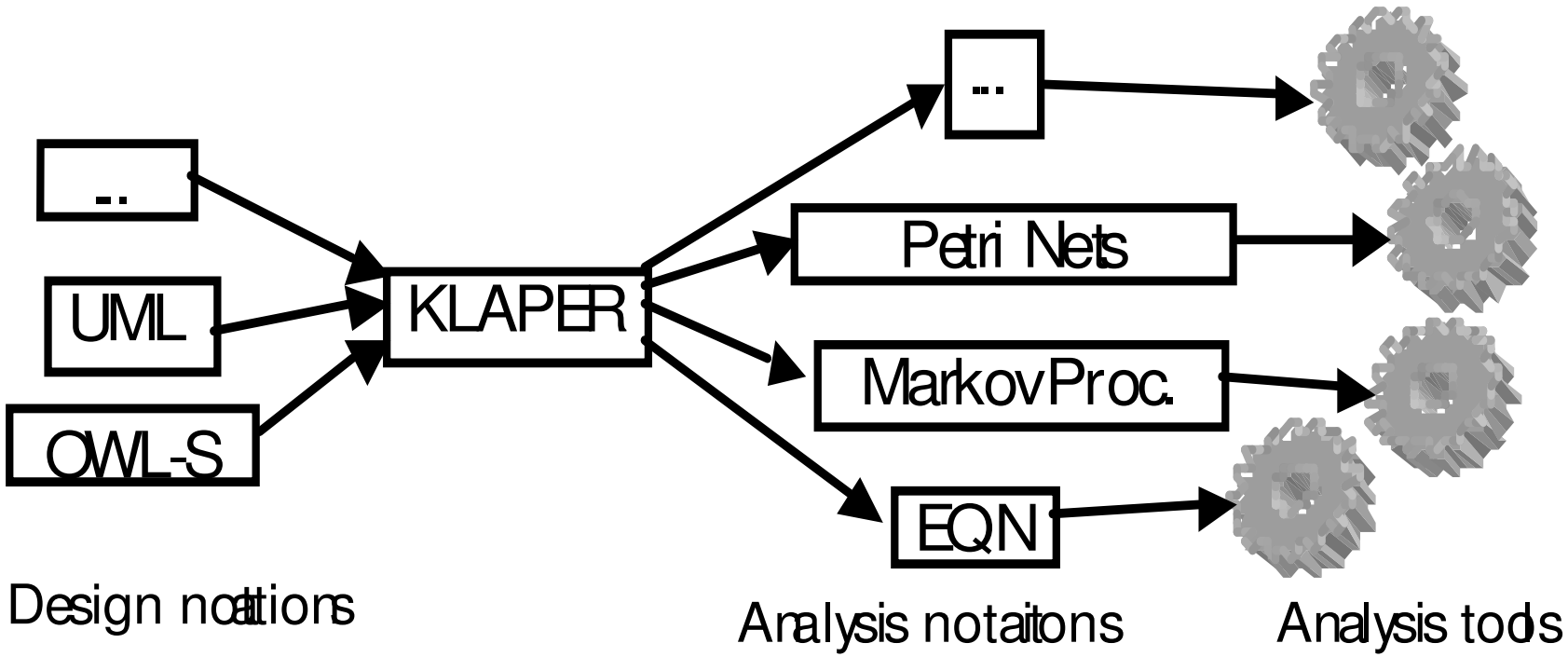
“tower of Babel” problem (1)

"tower of Babel" problem
- specific issue for CB systems
- heterogeneous component and composition description languages
  - acme, etc ...
- several target languages
  - queueing networks, Petri Nets, Markov Processes, Bayesian models, ...
- solution ?

"tower of Babel" problem



Intermediate languages: KLAPER



KLAPER

KLAPER : an intermediate "kernel" language to support performance and reliability analysis of CB systems based on model transformations

kernel : compact language (only CB design level information that is relevant for performance and reliability analysis is represented)

- transformation methodologies from/to design/analysis models made simpler, thanks to the reduced "semantic gap"

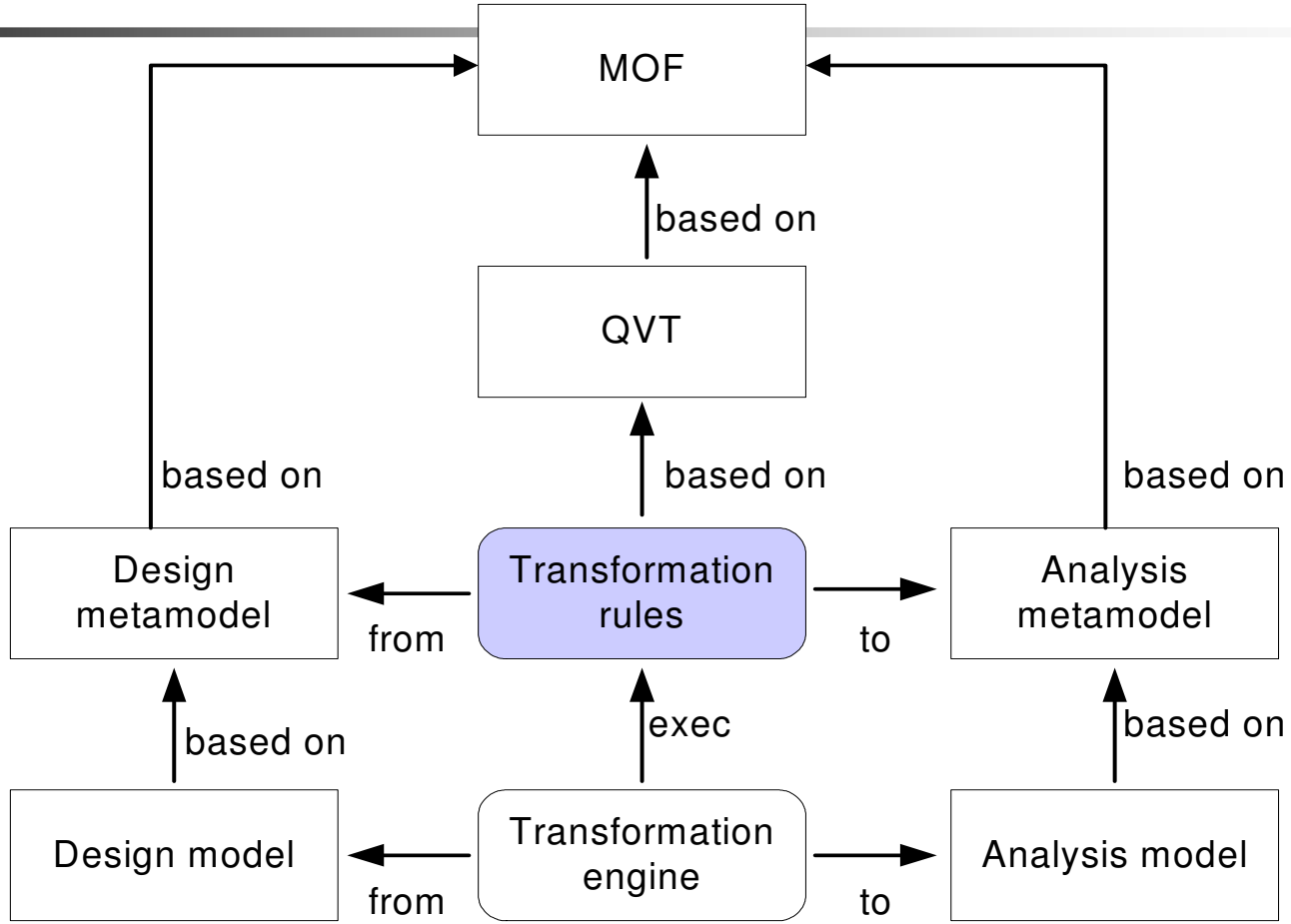
intermediate : neither a design language nor an analysis language

- from *MxN* to *M+N* transfomation methodologies

model transformation : MOF based, to exploit tools developed within *model-driven* approaches to software design (MDA)

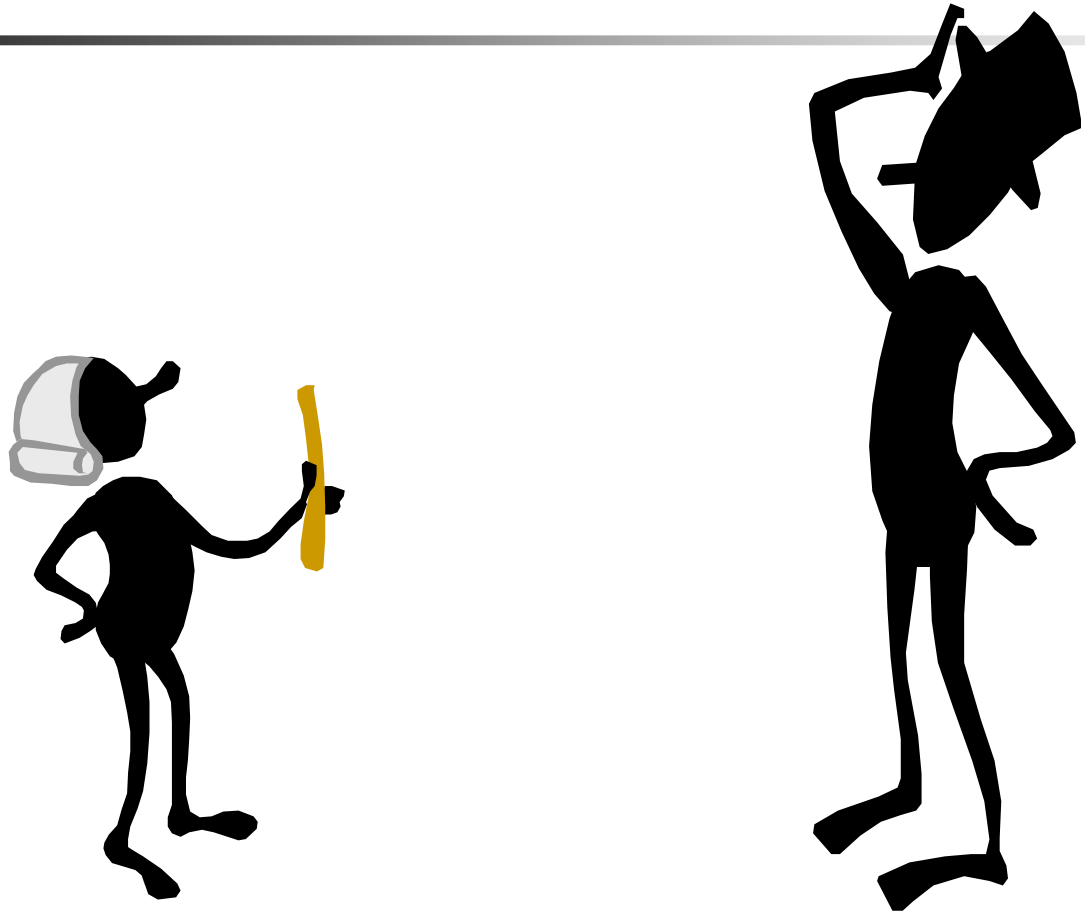
- transformation rules defined at metamodel level

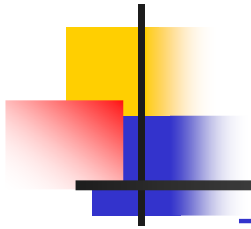
Shifting-upwards the focus



KLAPER is based on this phylosophy

Shifting-upwards the focus…





KLAPER: the basic concepts

The system modeled as a set of **Resources** and related **Services**

no basic distinction between *hw/sw*, *active/passive*, ...

Services are *offered* by Resources

Services can be *used* by other Services

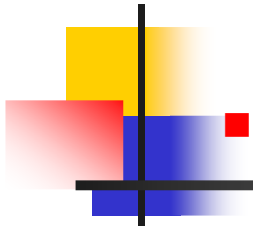
Services are characterized by an abstraction of their functional (constructive) behavior

Related concept : *analytical interfaces* vs. *constructive interfaces* for components (see *PECT* initiative from CMU-SEI)

KLAPER distills analytic interfaces from CB design models

… and models analogously platform “interfaces”

More about “distillation” (1)



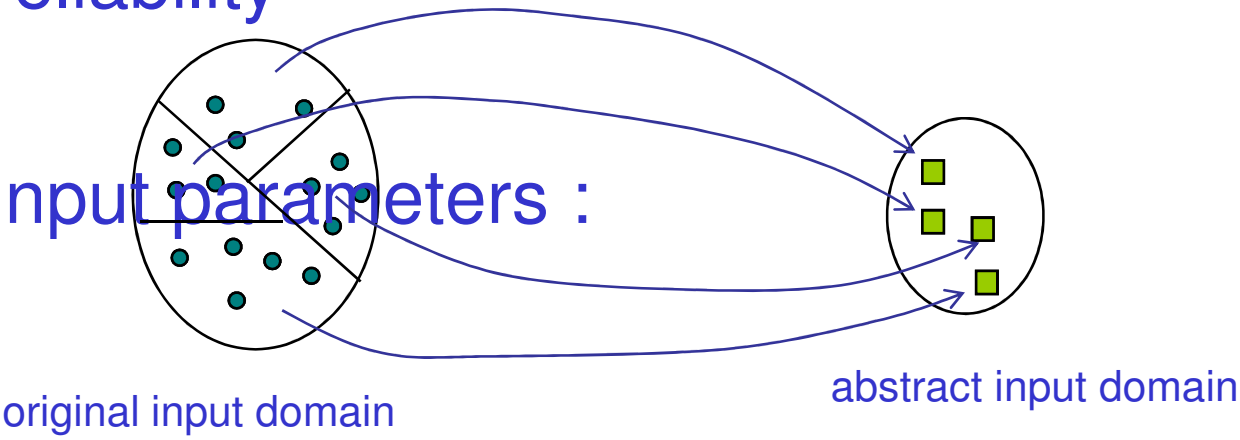
- "abstract" representation of an offered service
- what should be abstracted?

  the service behavior is dependent on :

  - input parameters used in a service invocation
- abstract representation of :
  - input parameters
  - behavior of other required services
  - flow of requests addressed to other components (and connectors)

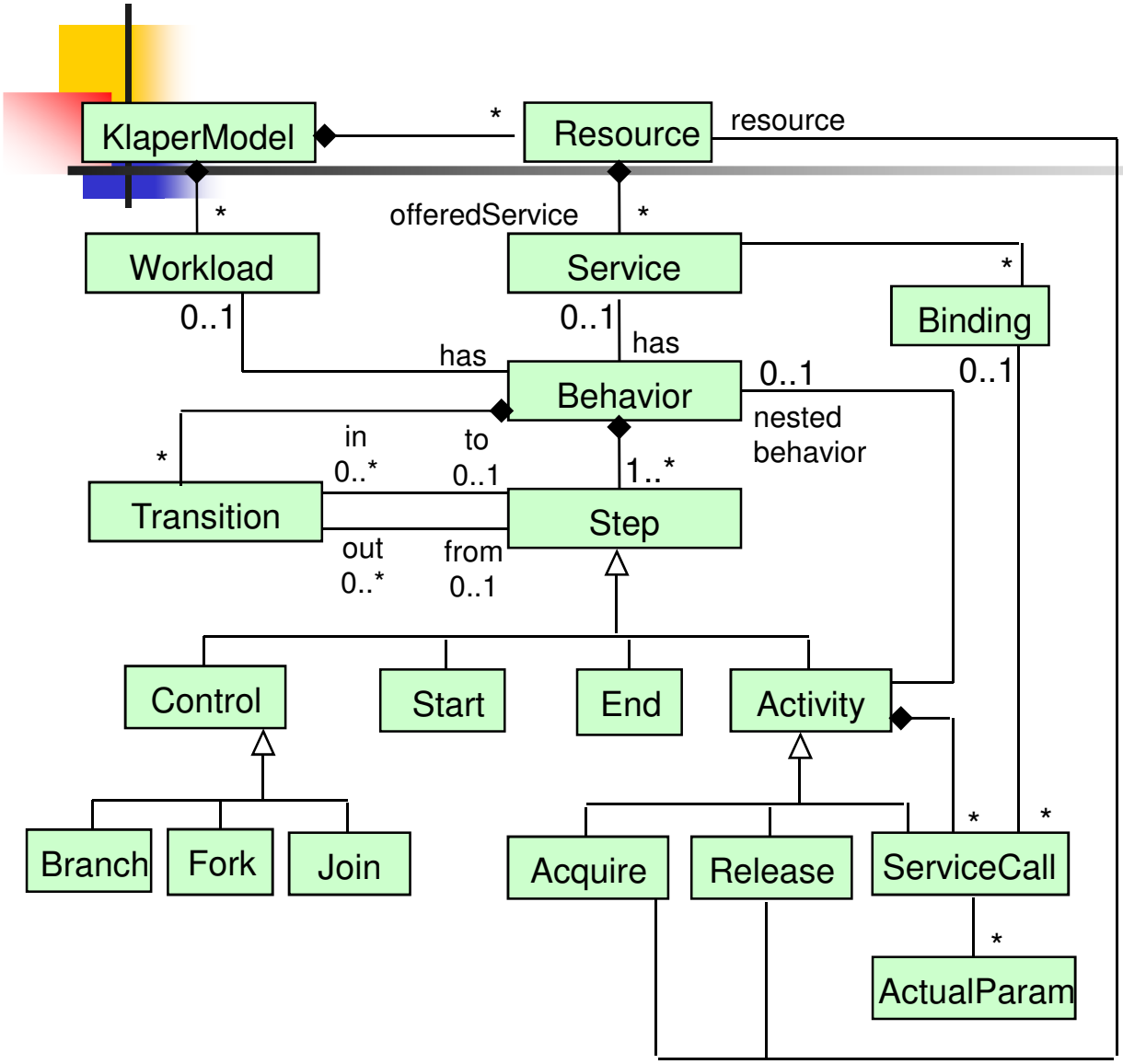
More about “distillation” (2)

- stochastic abstractions to support stochastic analysis of performance and reliability
- input parameters :

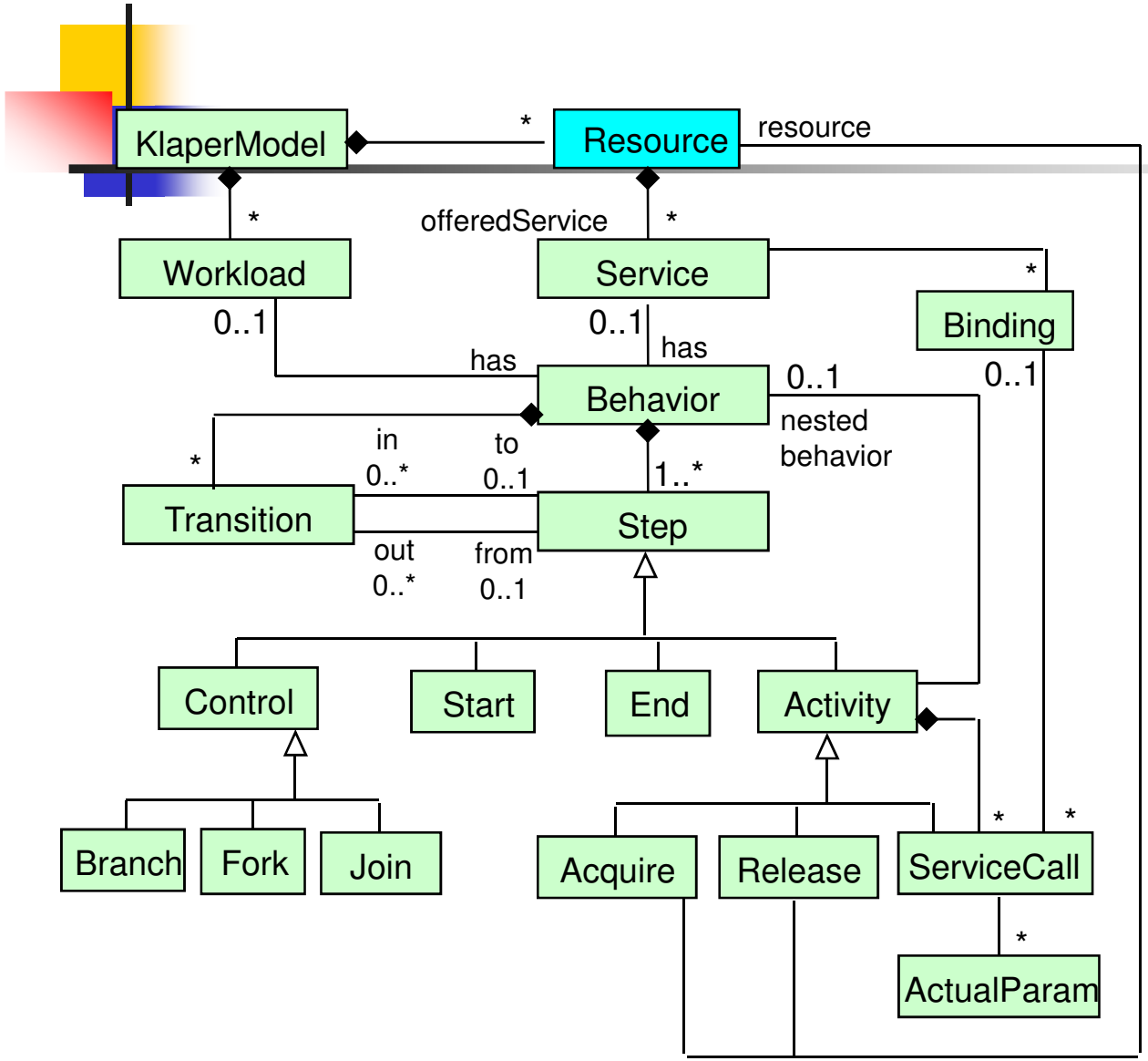


- flow of requests :
  - probabilistic representation of :
    - flow of control (modeled by probabilistic branching and loops)
    - actual parameters (modeled by random variables)

The KLAPER metamodel



The KLAPER metamodel



Resource

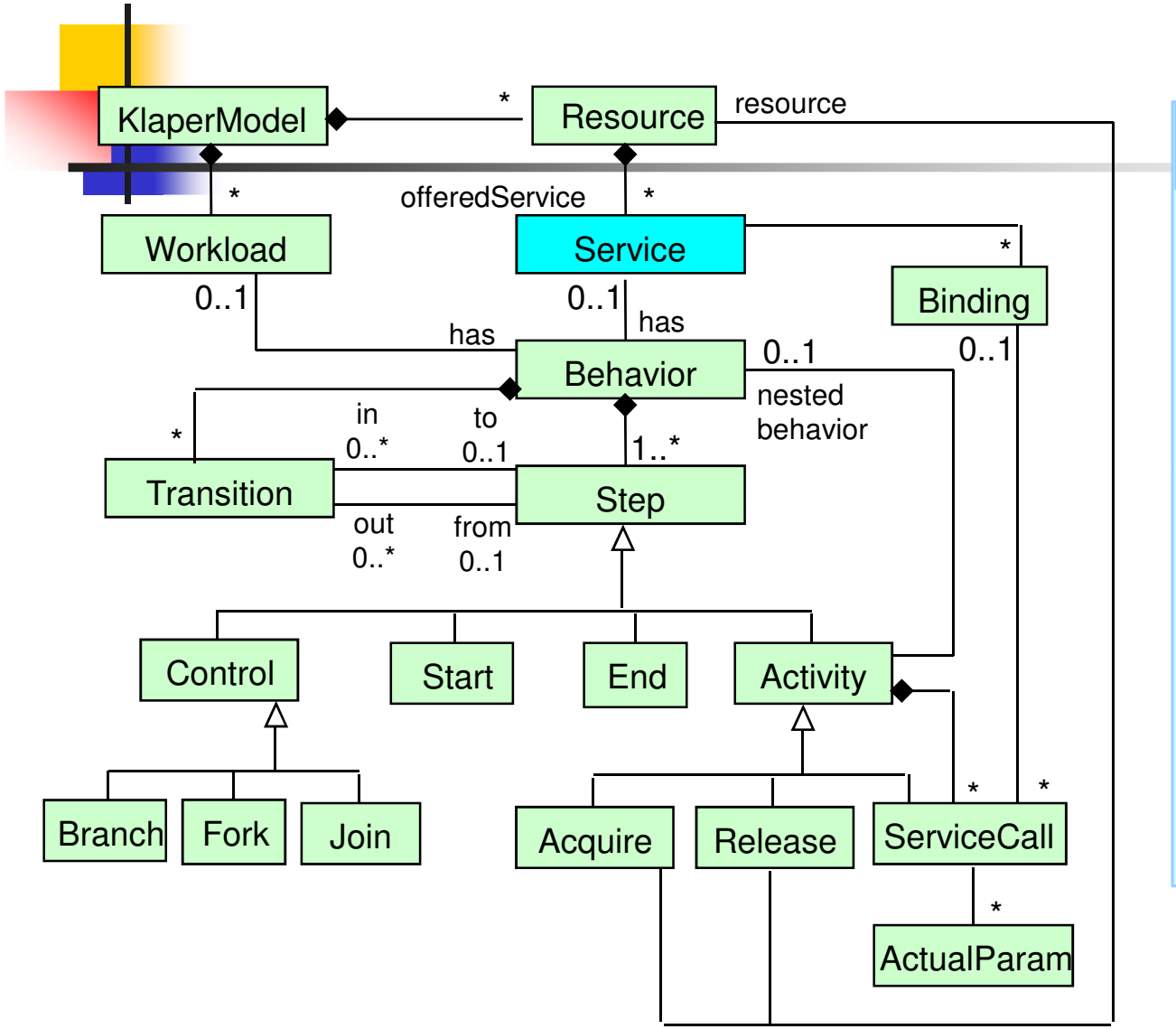
Attributes:

- name
- type
- capacity
- schedPolicy
- description

Associations

- offeredService

The KLAPER metamodel



Service

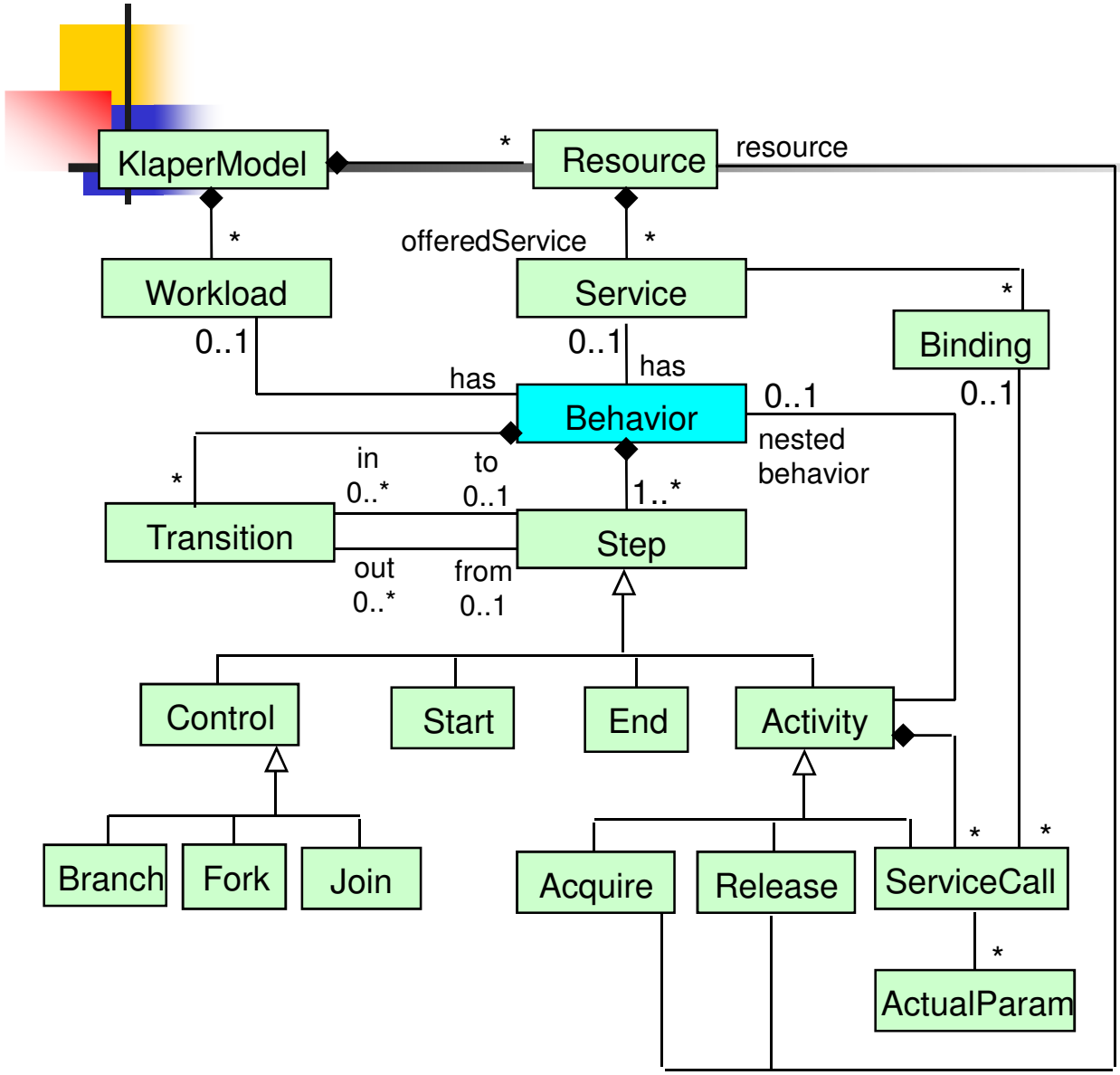
Attributes:

- name
- formalParams
- speedAttr
- failAttr
- description

Associations

- behavior
- resource
- binding

The KLAPER metamodel

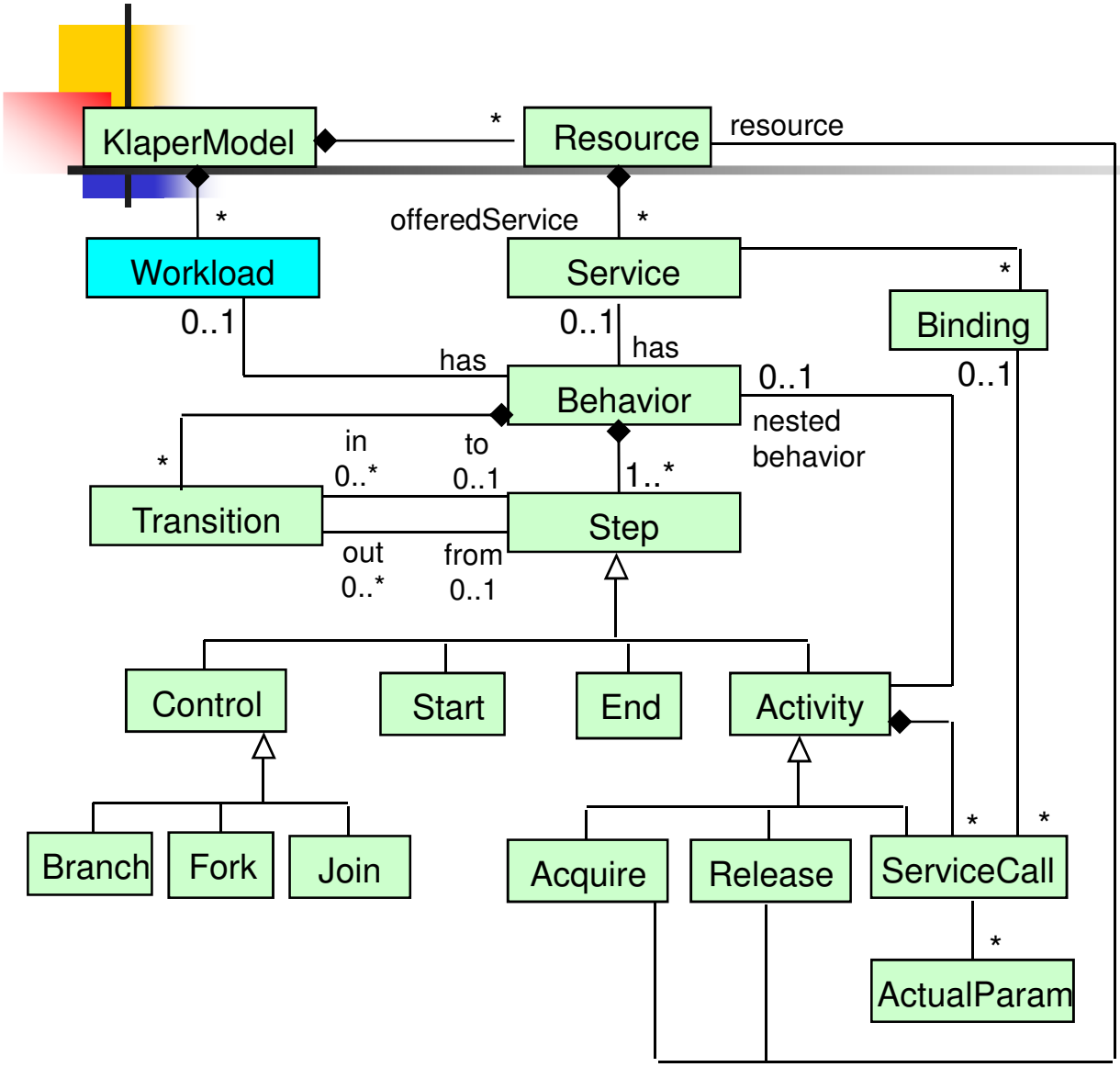


Behavior

Associations

- step
- transition
- service
- workload

The KLAPER metamodel



Workload

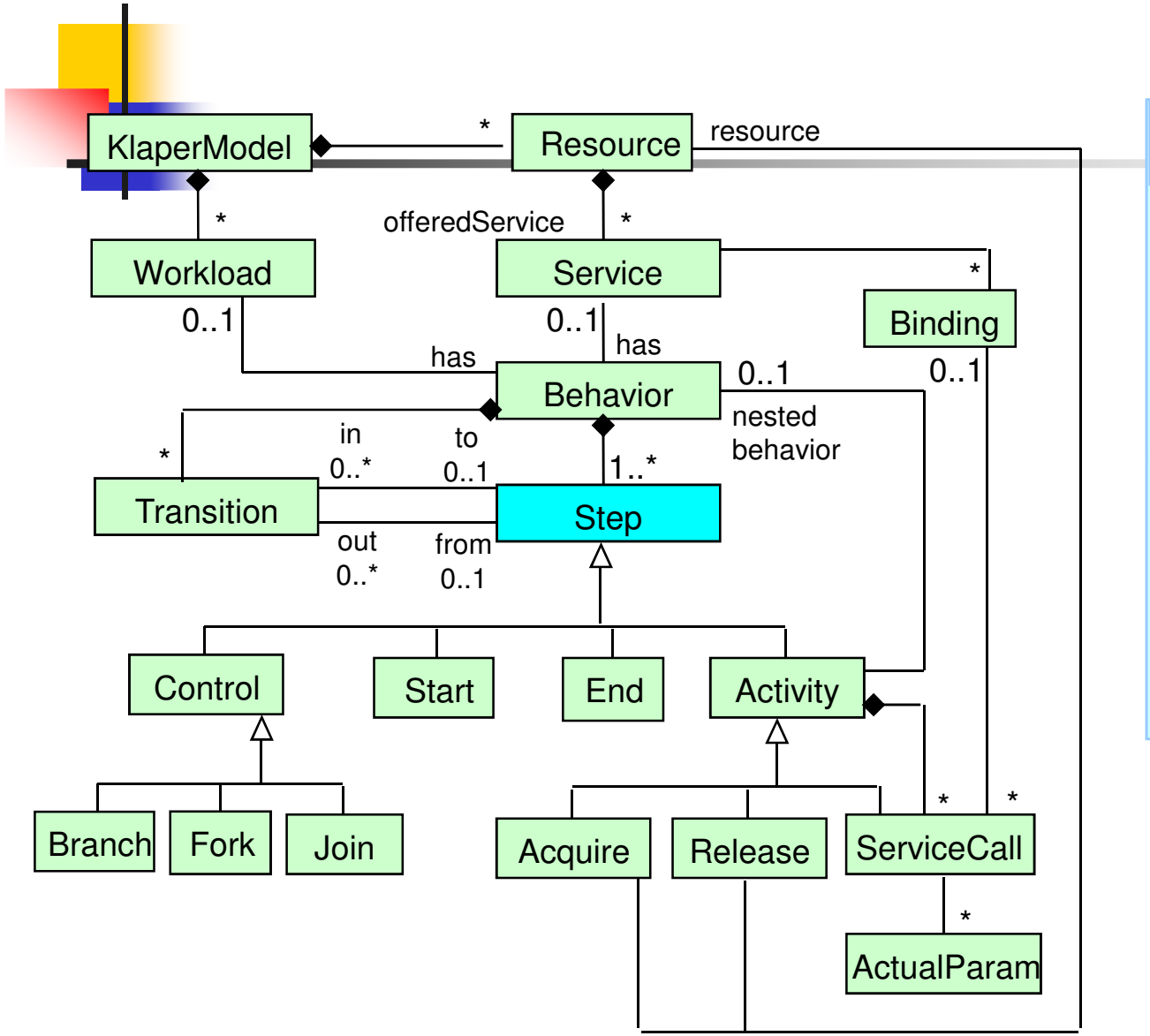
Attributes:

- workloadType
- arrivalProcess
- population

Associations

- behavior

The KLAPER metamodel



Step

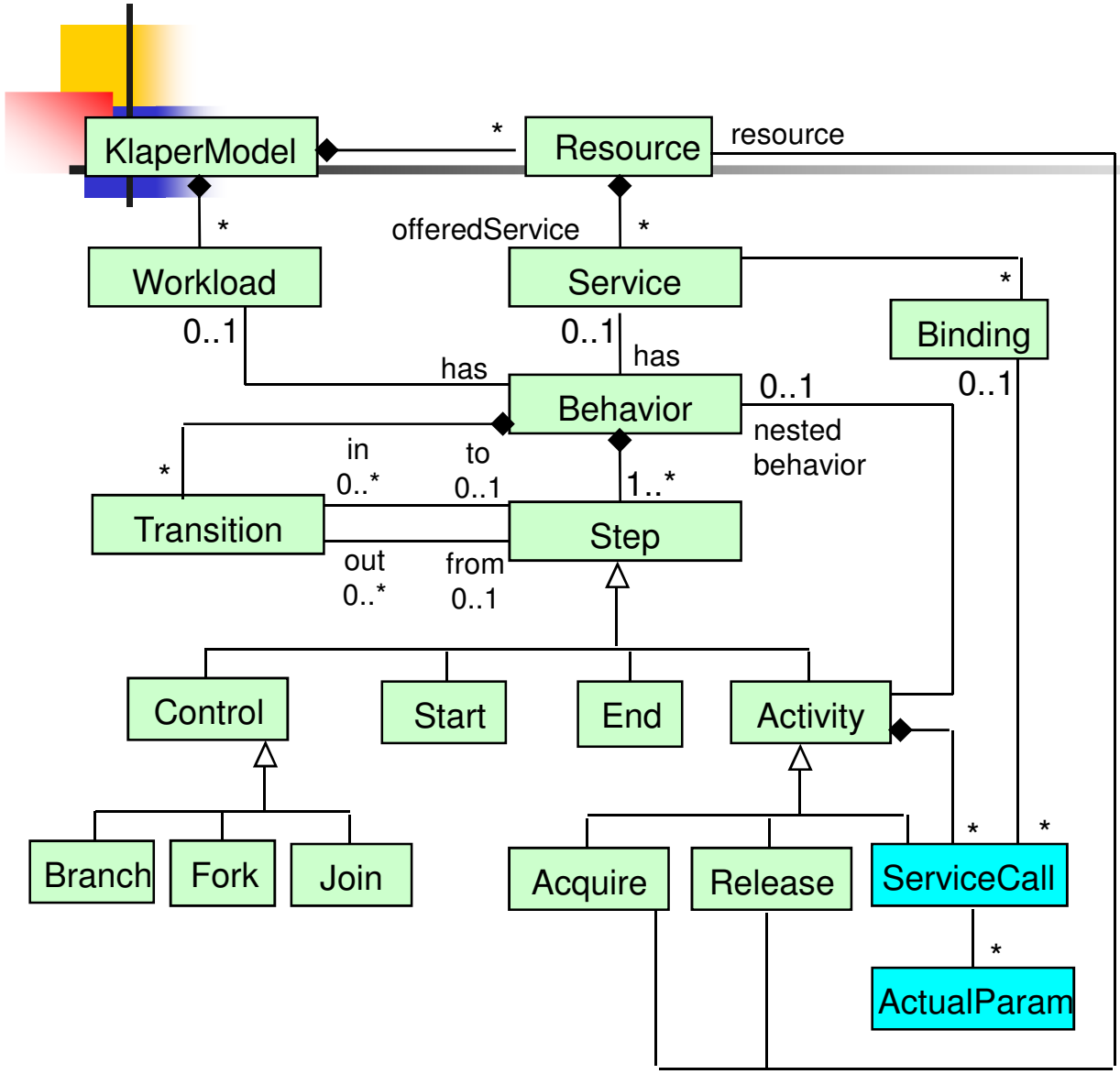
Attributes:

- name
- repetition
- internalExecTime
- internalFailProb
- completionModel

Associations

- transition

The KLAPER metamodel



ServiceCall

Attributes:

- resourceType
- serviceName
- isSync

Associations

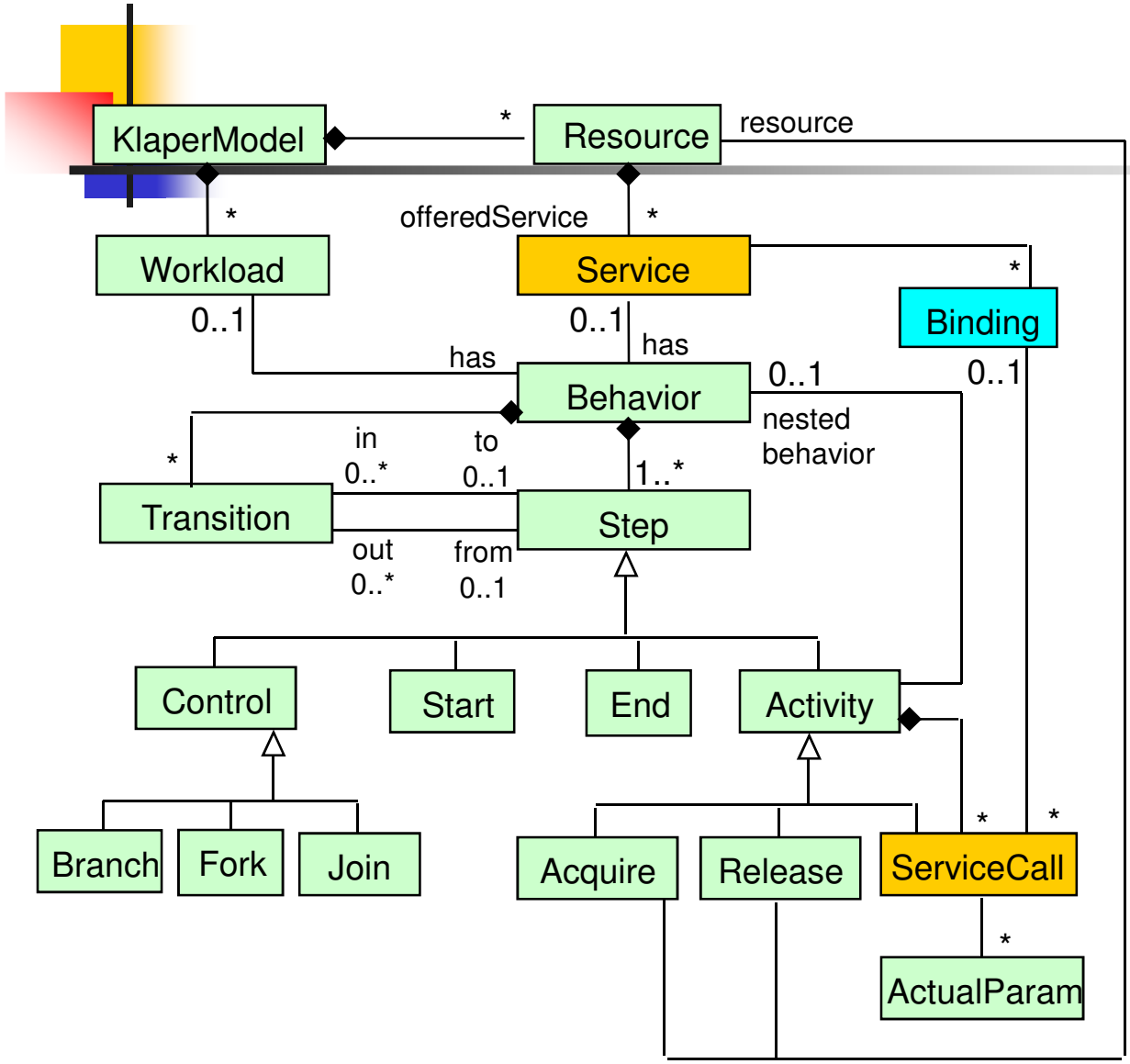
- activity
- actualParam
- binding

ActualParameter

Attributes:

- value

The KLAPER metamodel

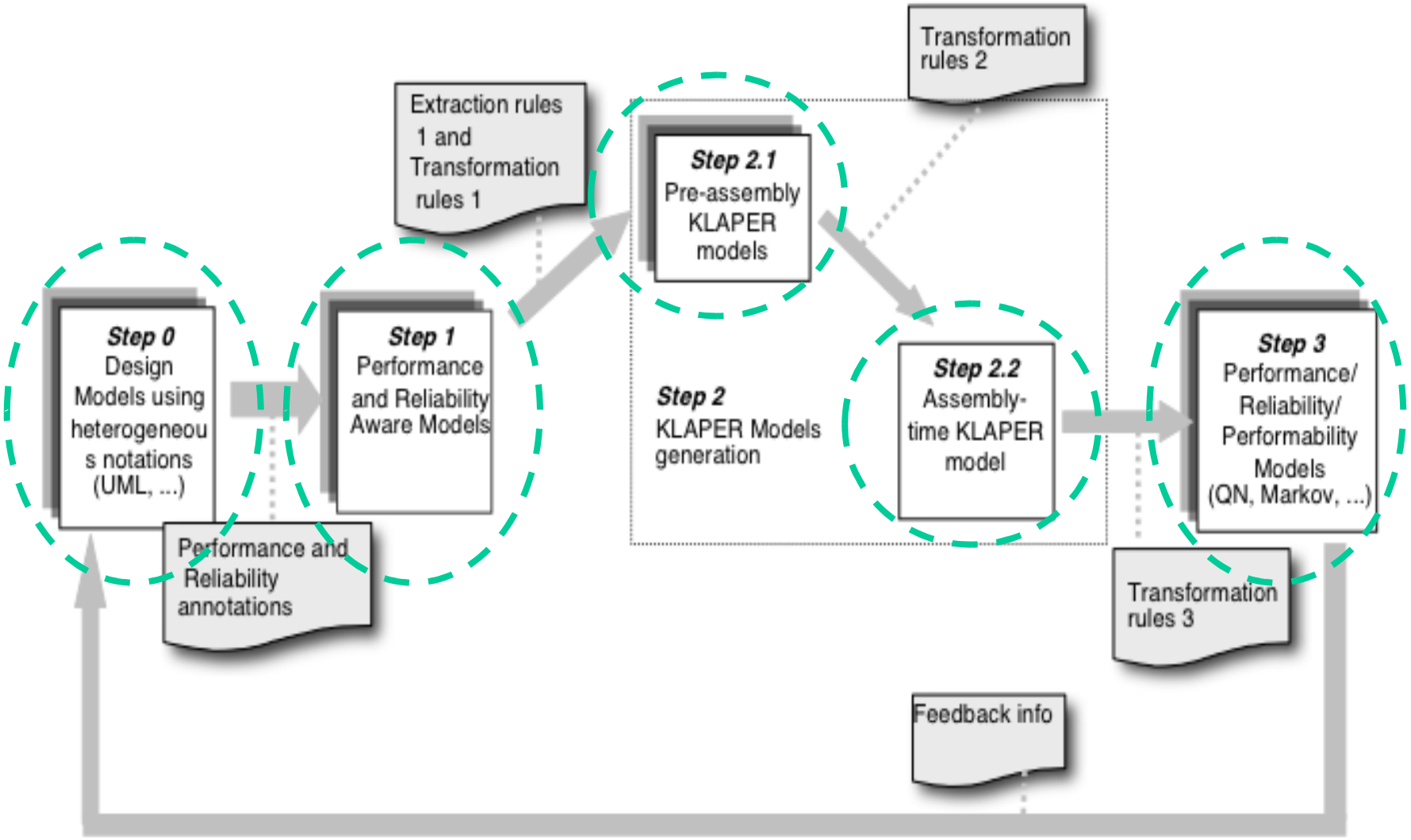


Binding

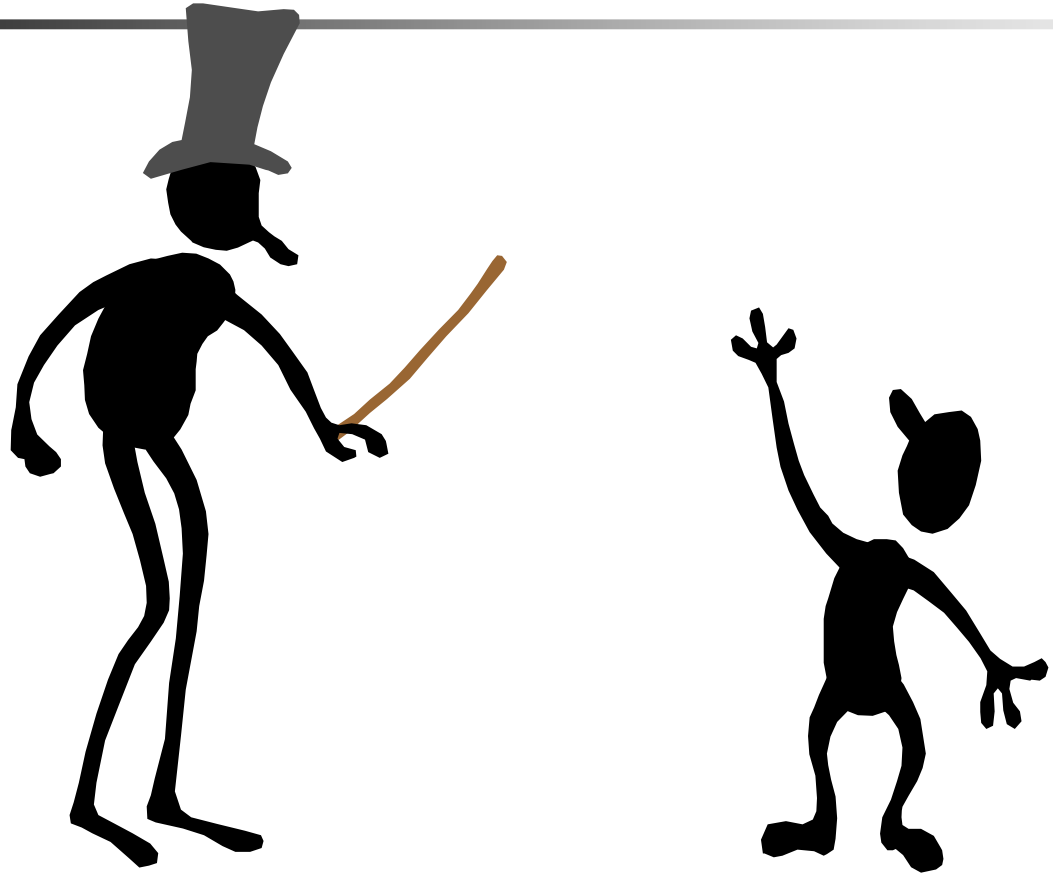
Associations:

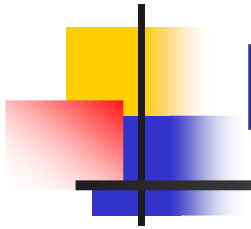
- ServiceCall
- Service

KLAPER-based model generation



Shifting again…



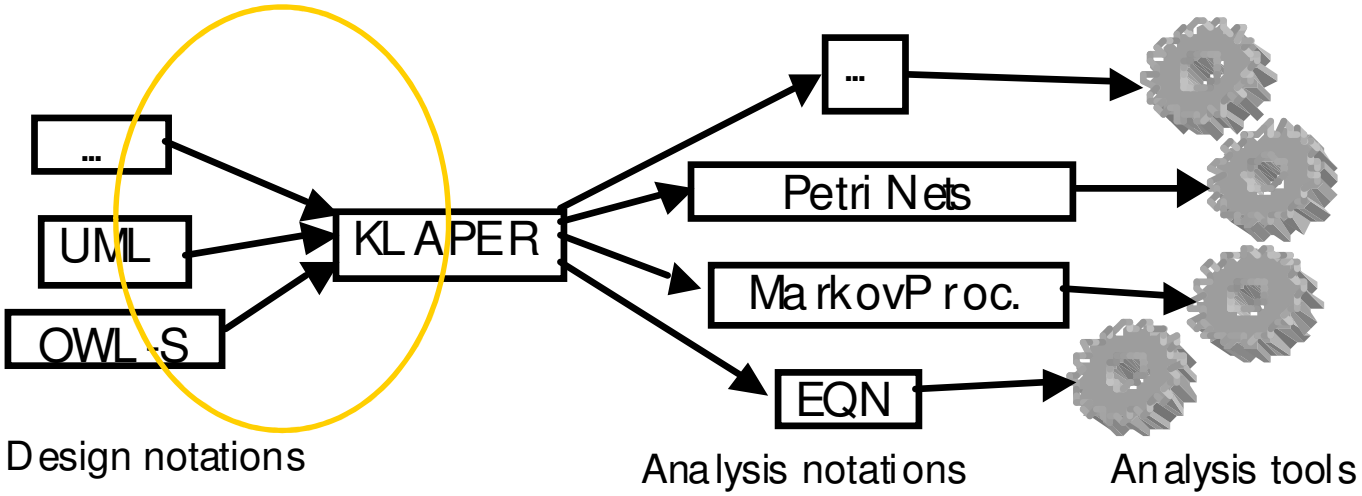


Modelling CoCoME

- What do we model?
- UC1 – Process Sale
  - Bar Payment only (Card Payment not modeled)
  - Main flow only (not secondary or exception flows)
- UC3 – Order Products
  - Completely modeled

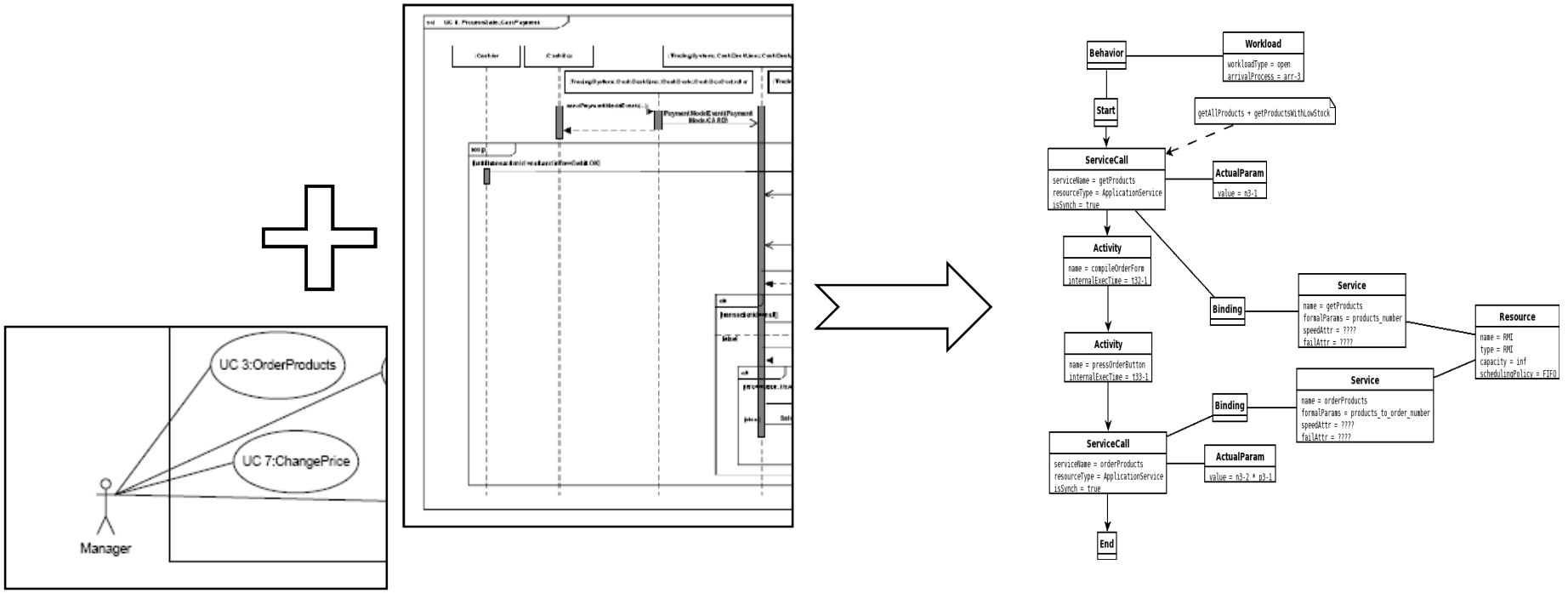
From design models to KLAPER models

- A case study: from UML to KLAPER.
  - examples of MDA-based transformations from a particular design model notation to KLAPER.



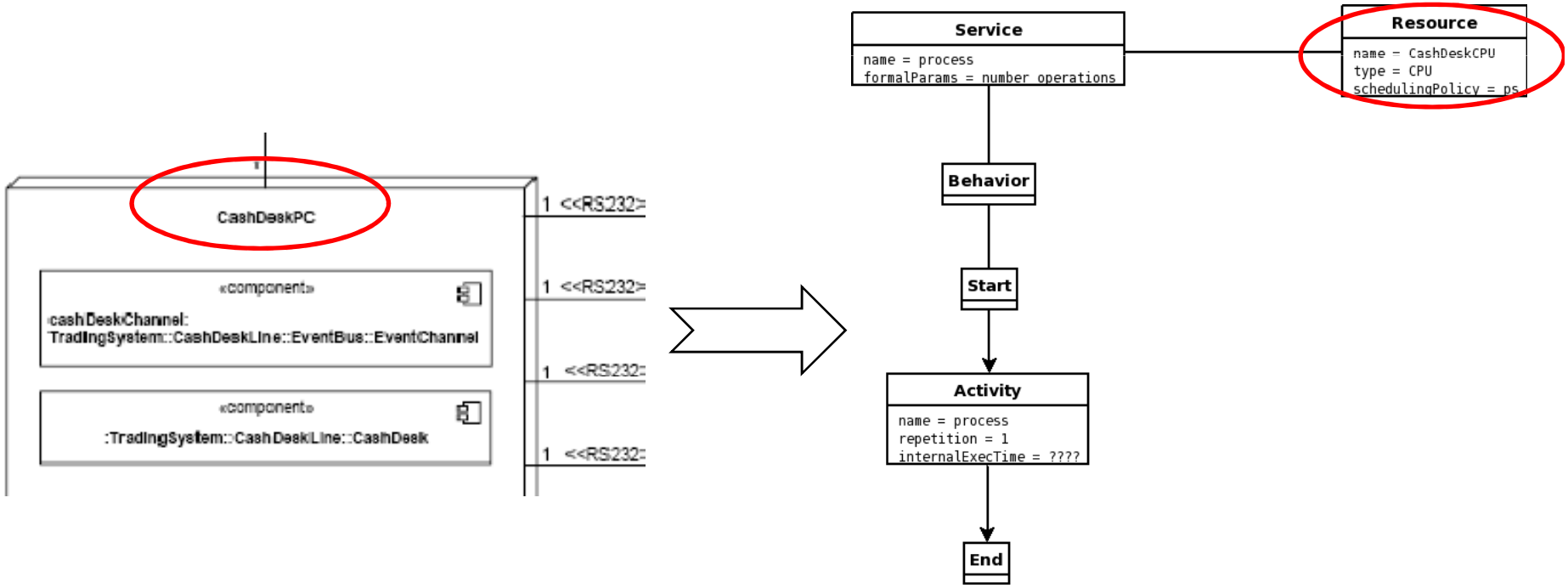
Transformation rules from UML to KLAPER

- Each UML use case is mapped onto a KLAPER workload whose steps are taken from the corresponding sequence diagrams.



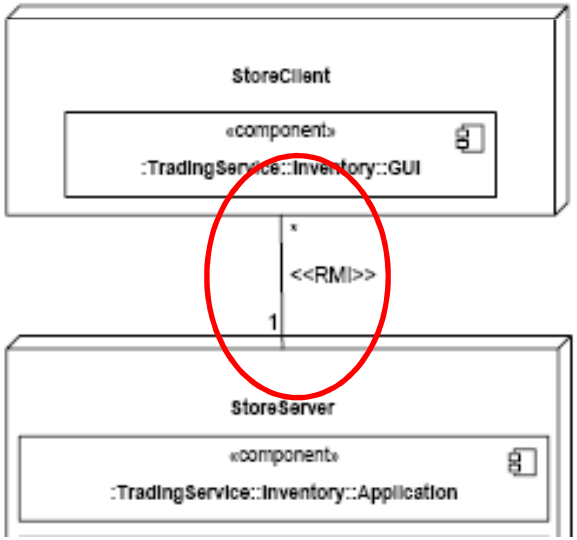
Transformation rules from UML to KLAPER

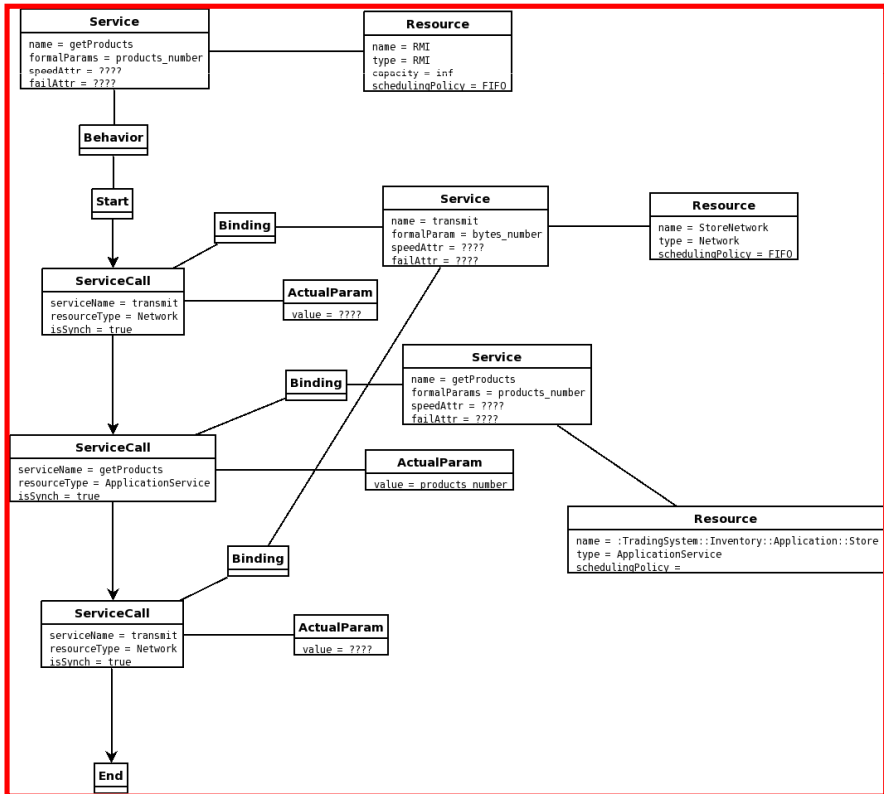
- KLAPER Resources are built from the UML deployment diagram.

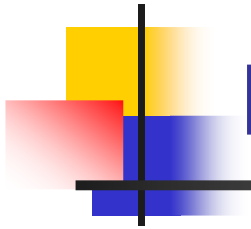


Transformation rules from UML to KLAPER

- Network connections of the UML deployment diagram are mapped onto KLAPER resources offering “connection services”.





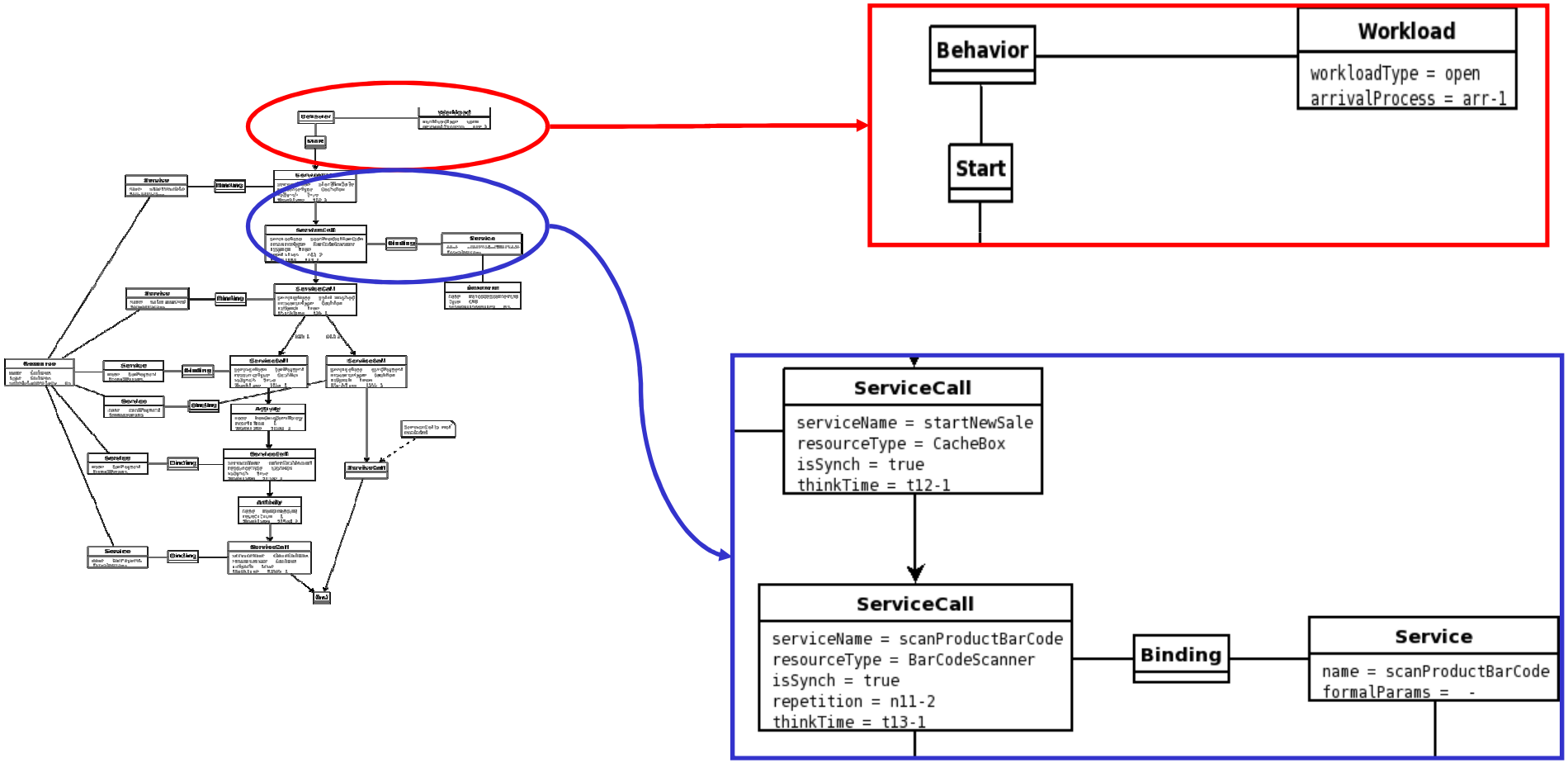


Transformation rules from UML to KLAPER

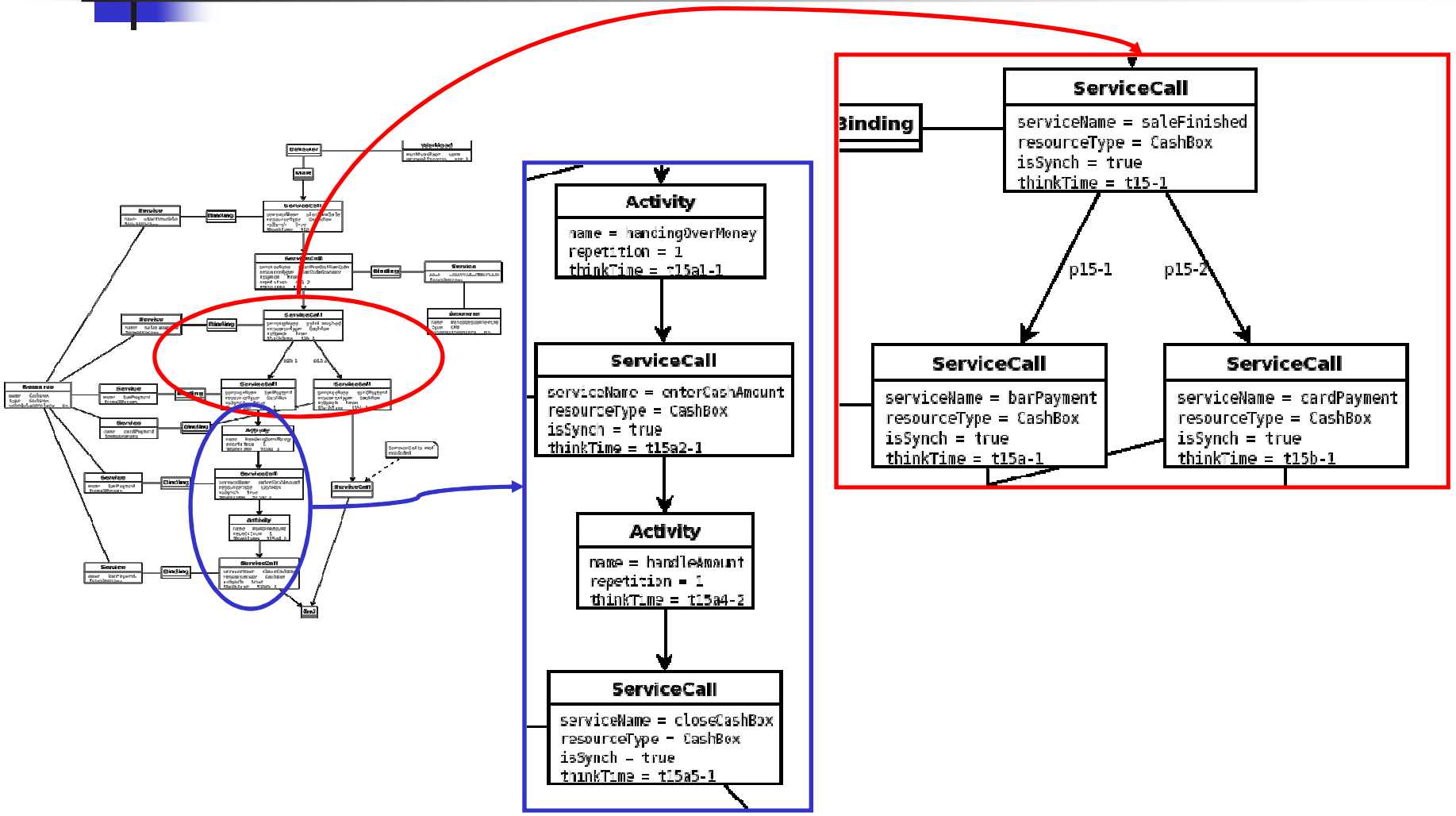
- These are only some examples of the transformation rules from UML to KLAPER.
- There are many other transformation rules!

(but it would take too much time to see all of them)

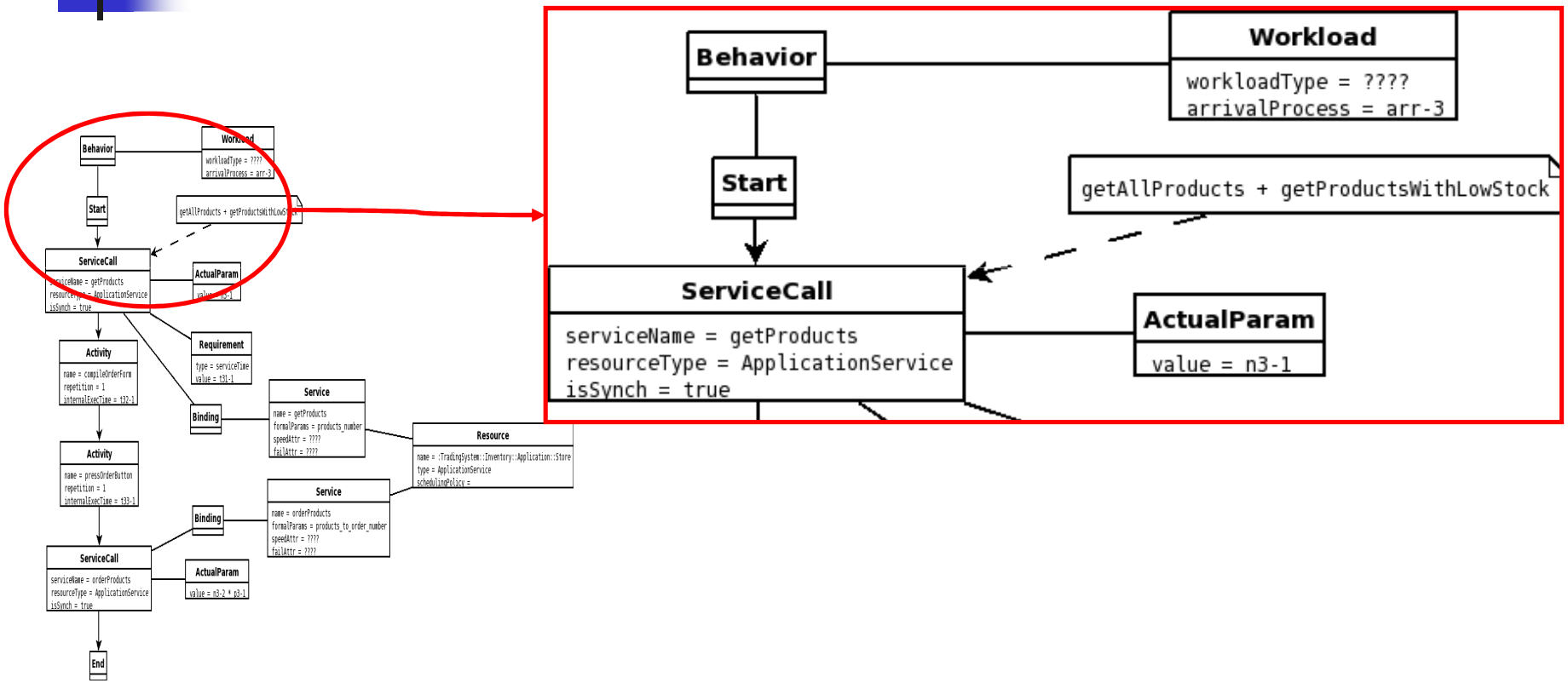
A KLAPER Workload modelling the UC1 main operation



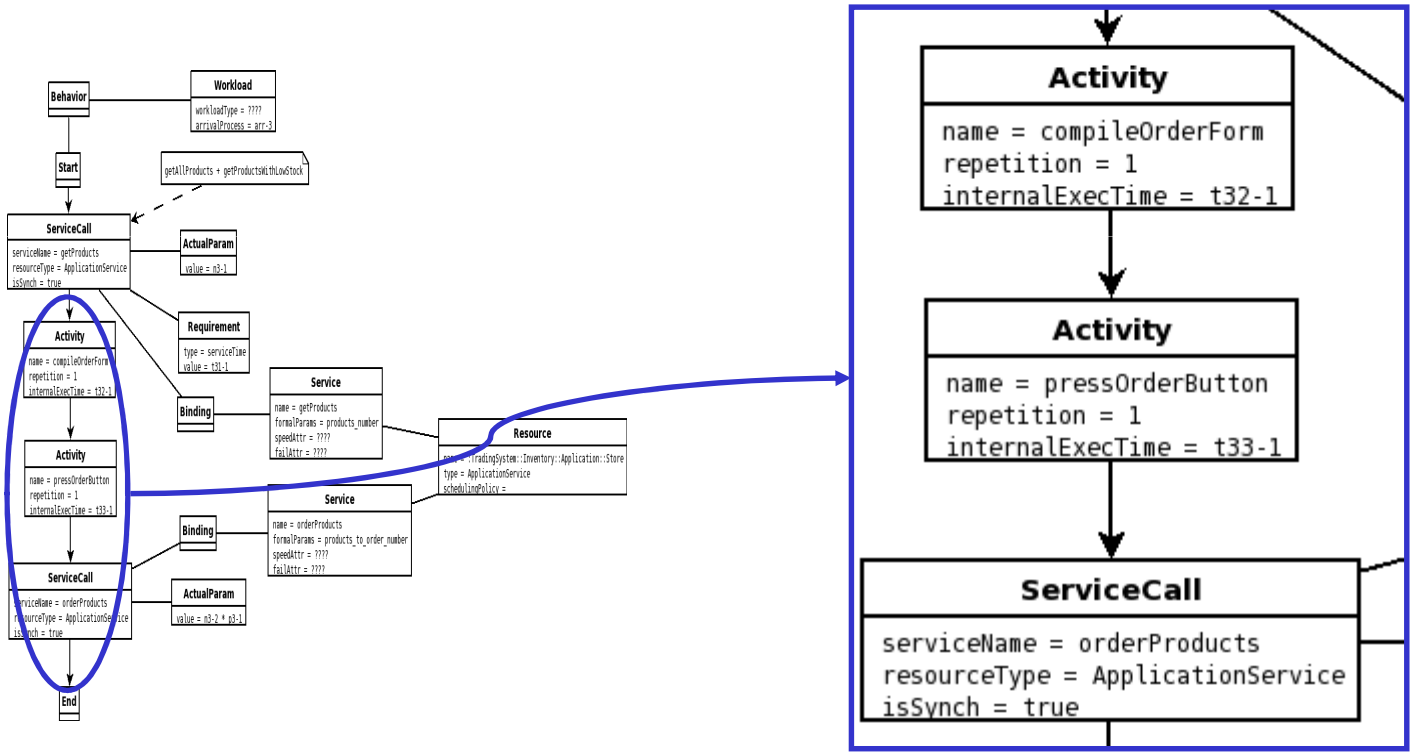
A KLAPER Workload modelling the UC1 main operation



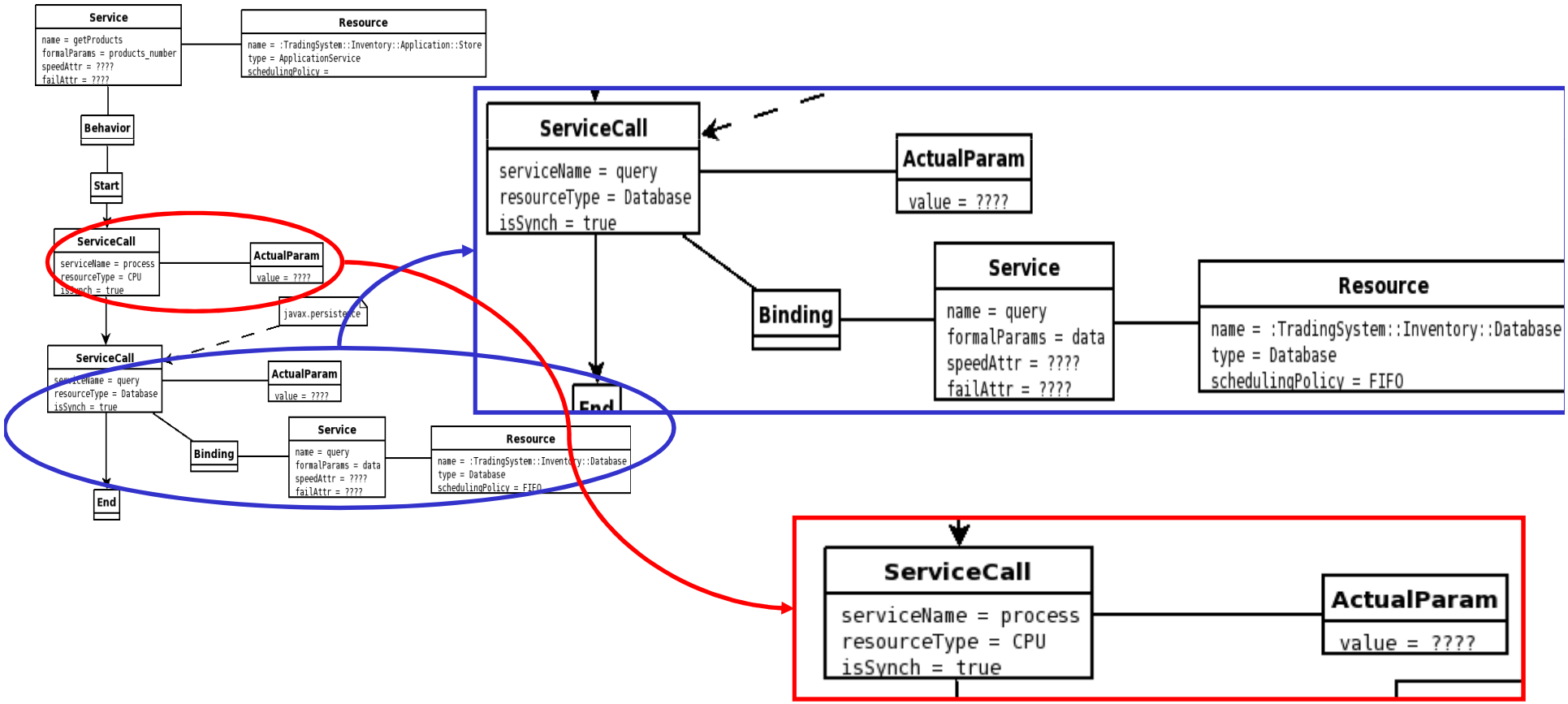
A KLAPER Workload modelling the UC3 main operation



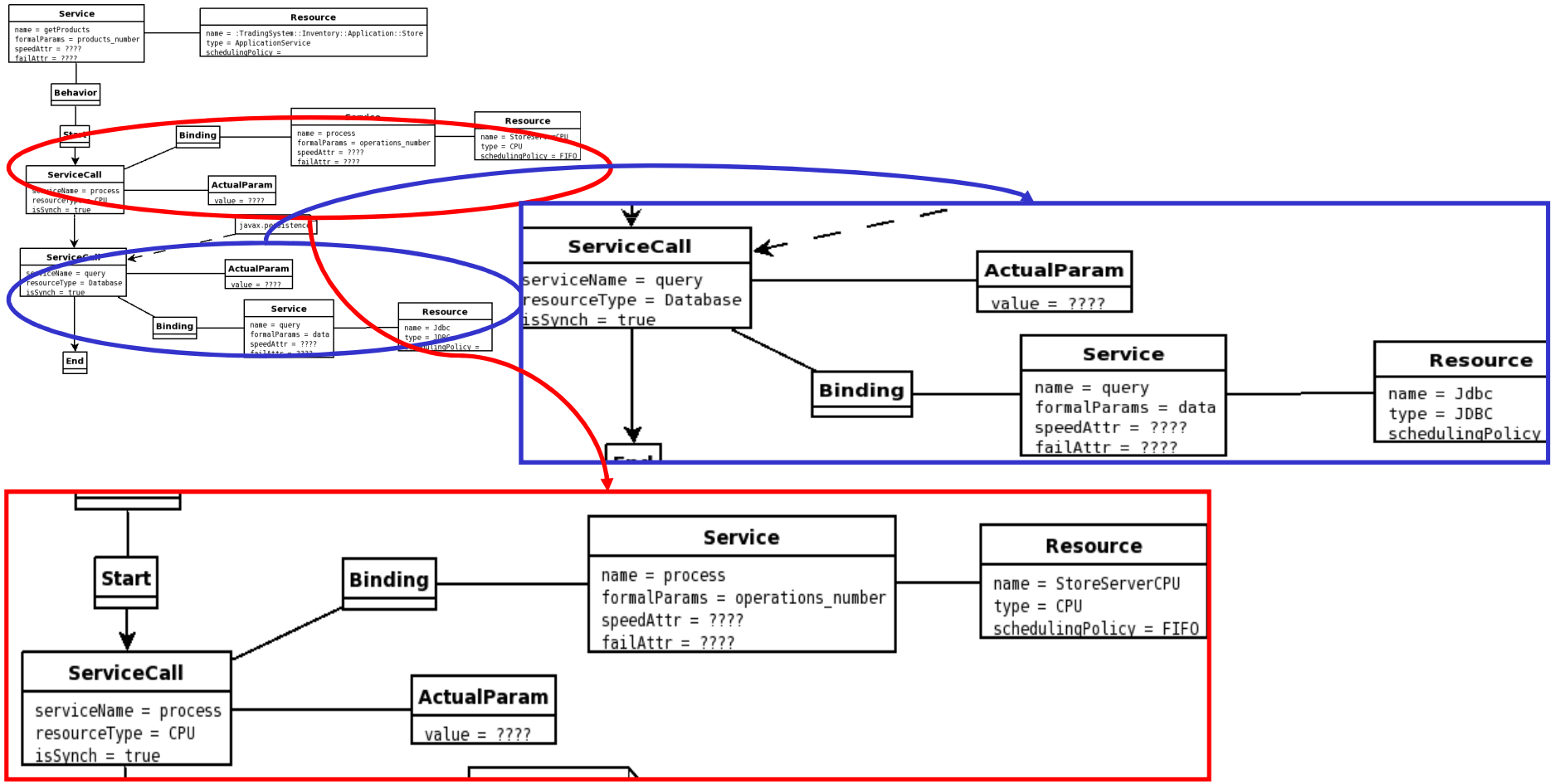
A KLAPER Workload modelling the UC3 main operation



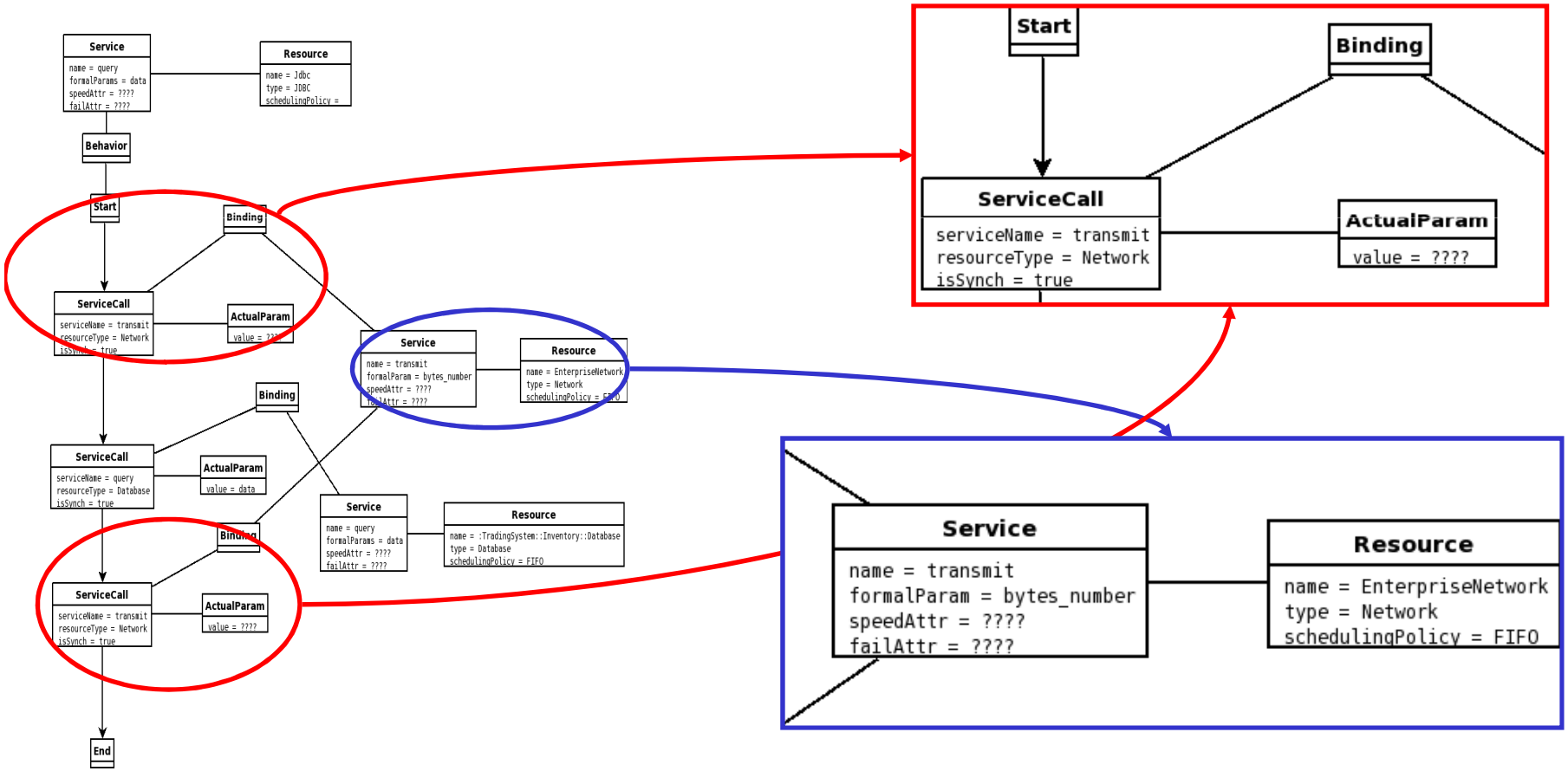
KLAPER model of the UC3 Store getProducts operation (pre-deployment)



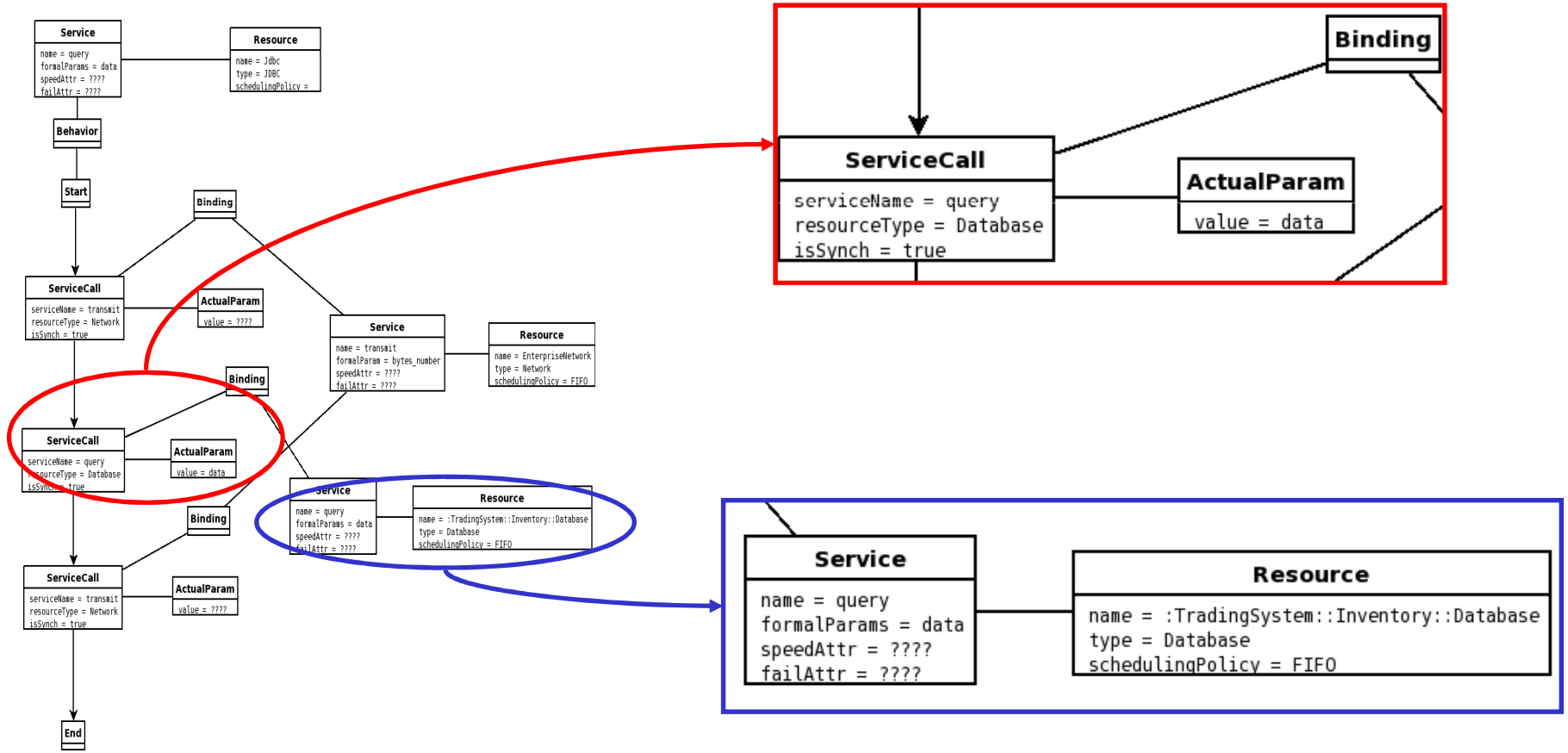
KLAPER model of the UC3 Store getProducts operation (post-deployment)



KLAPER model of the JDBC Resource

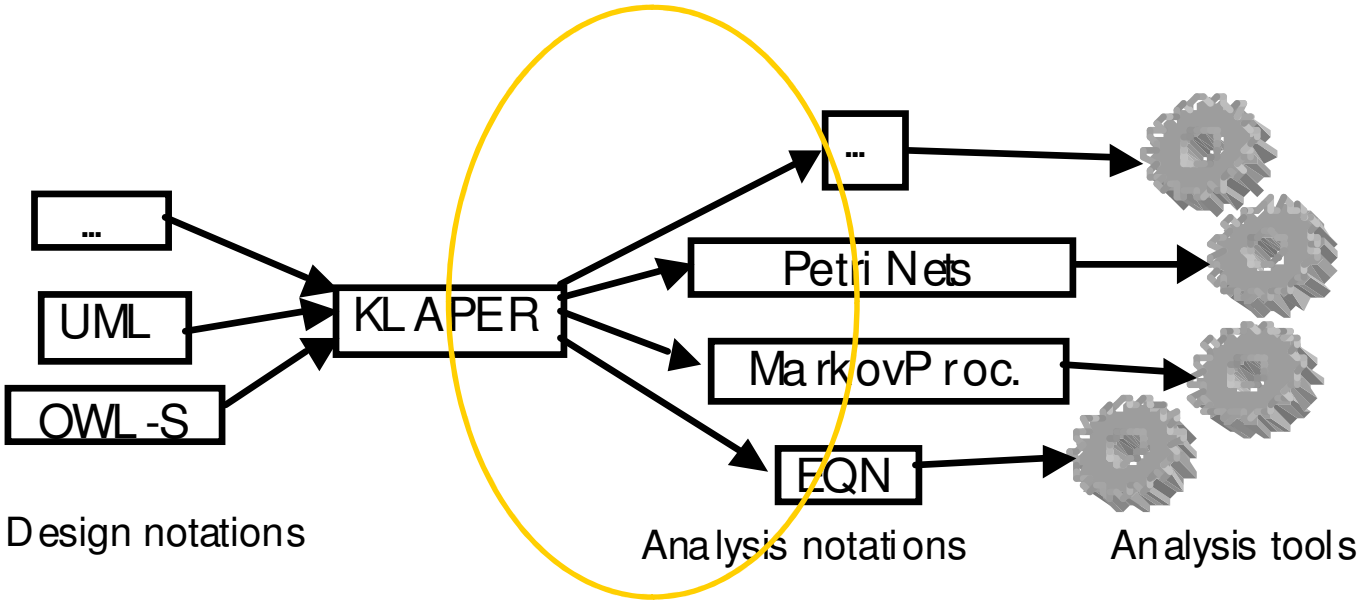


KLAPER model of the JDBC Resource



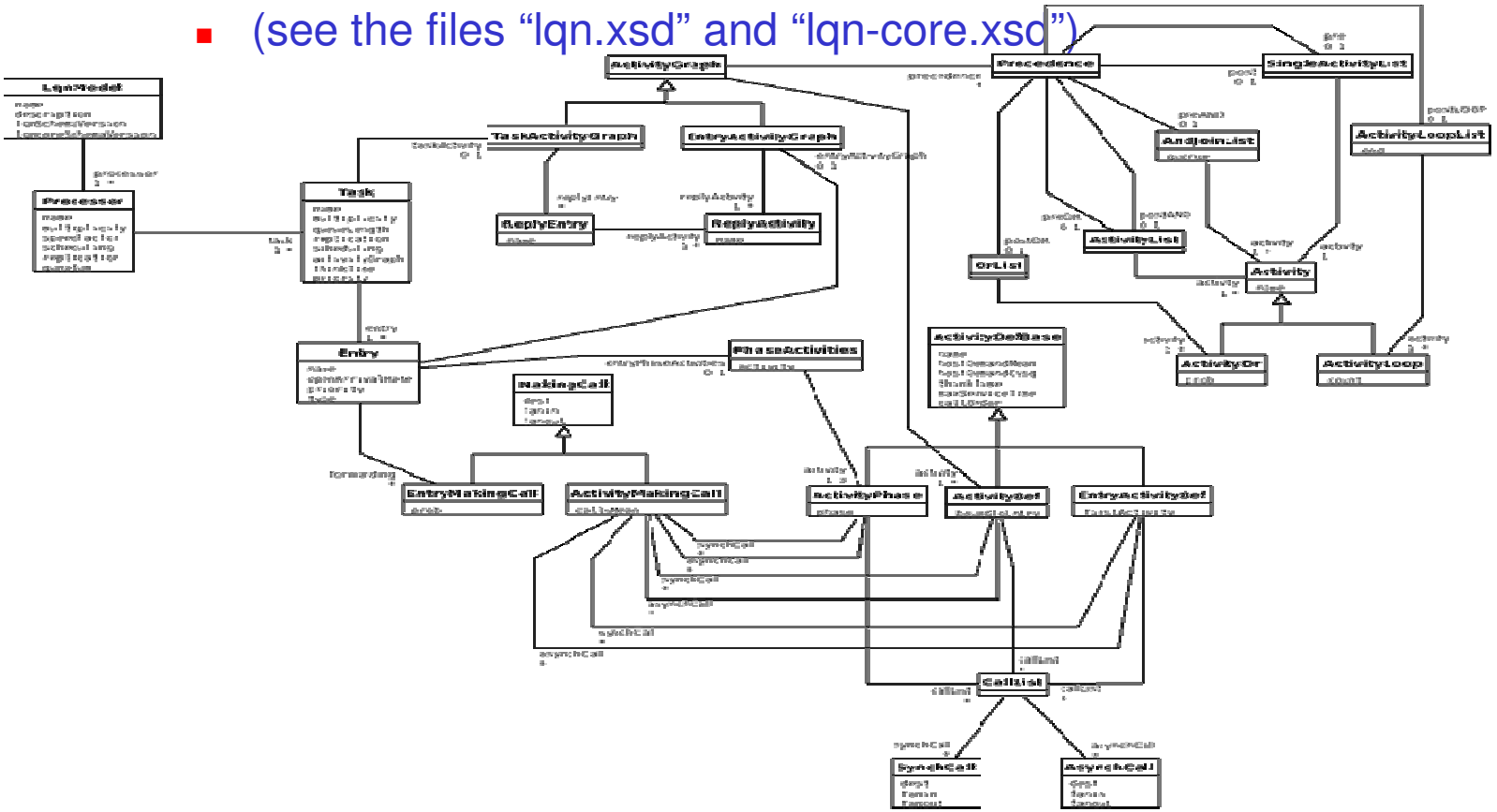
From KLAPER models to analysis models

- A case study: from KLAPER to LQN.
  - examples of MDA-based transformations from KLAPER to a particular analysis notation.



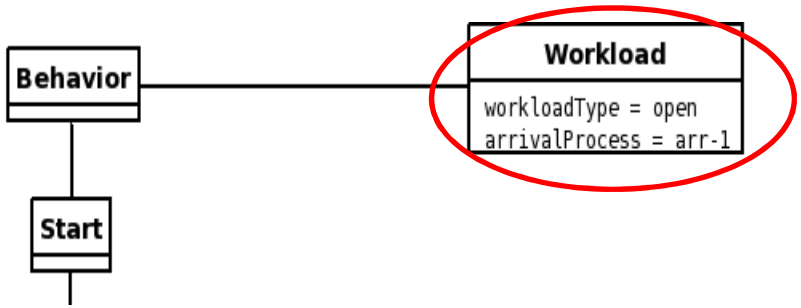
The LQN MOF Meta-Model

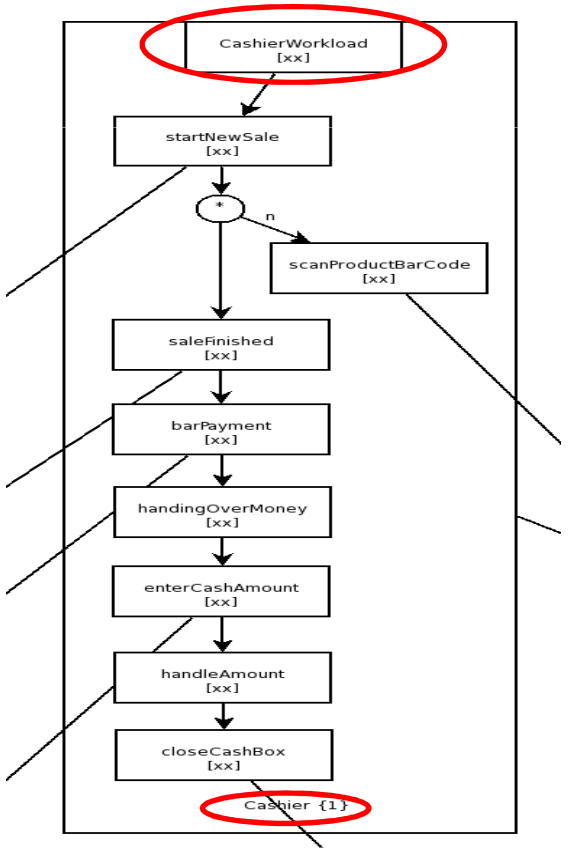
- Extracted directly from the XMLSchema used by the Carleton University *lqns* simulator
  - (see the files “lqn.xsd” and “lqn-core.xsd”)



Transformation rules from KLAPER to LQN

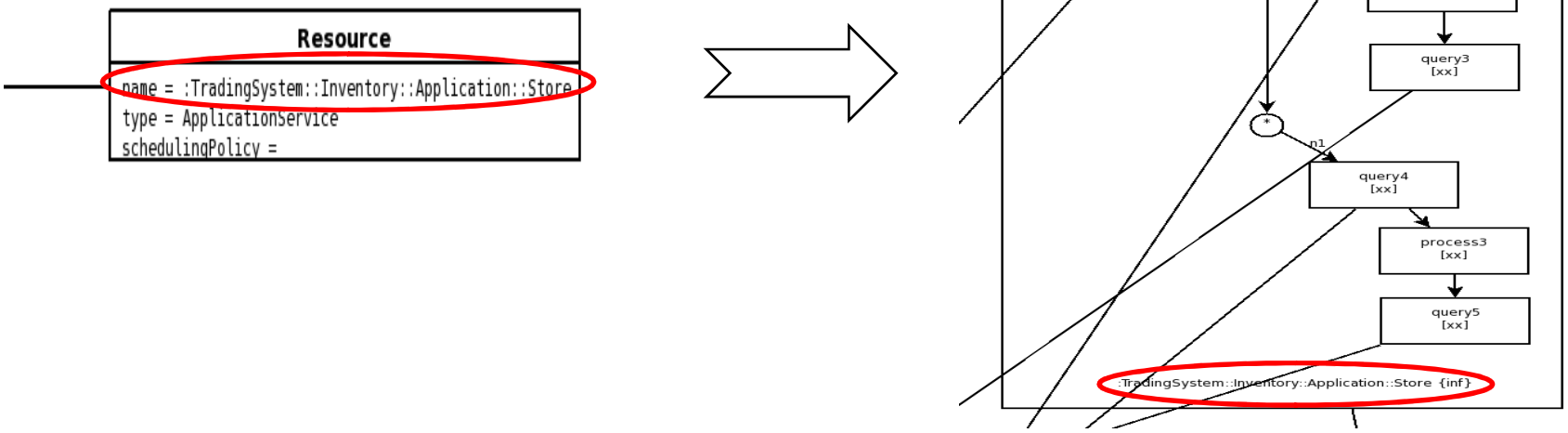
- Each KLAPER workload is mapped onto an LQN Task with an Entry where the workload type is specified.





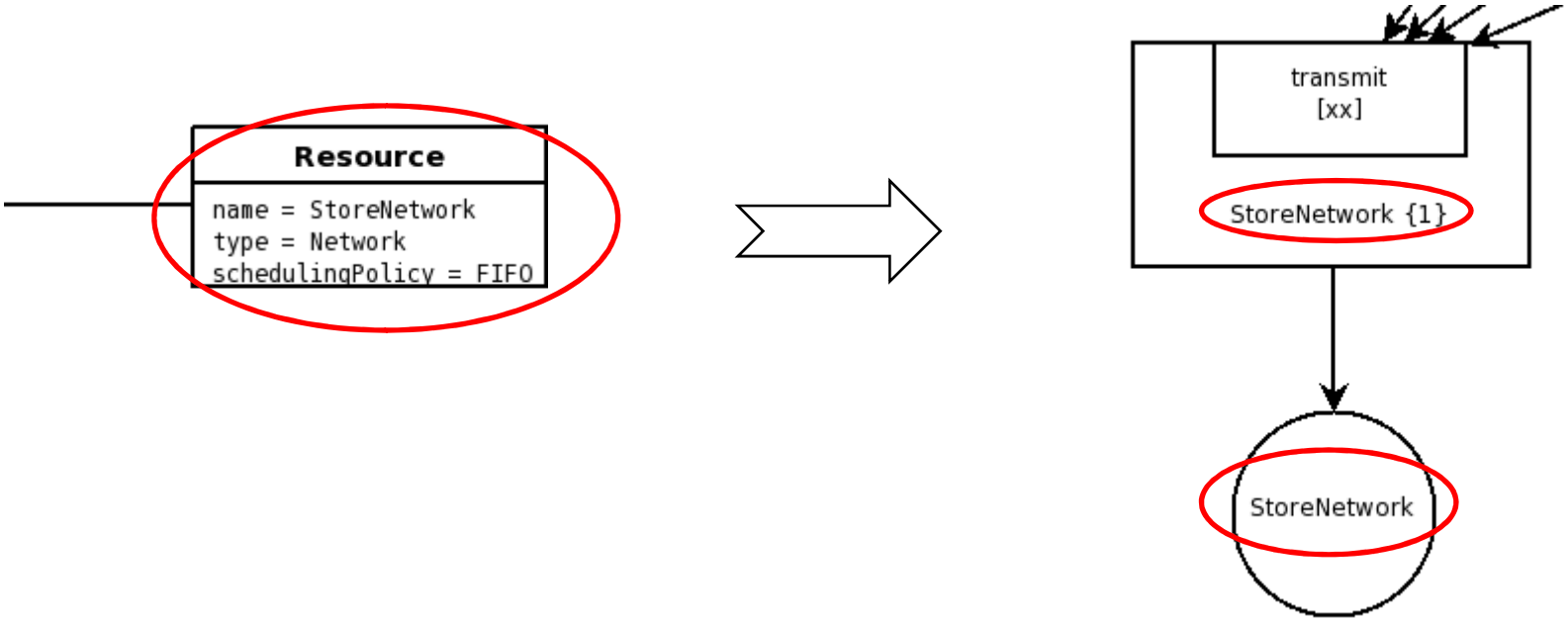
Transformation rules from KLAPER to LQN

- Each KLAPER Resource is mapped onto an LQN Task…



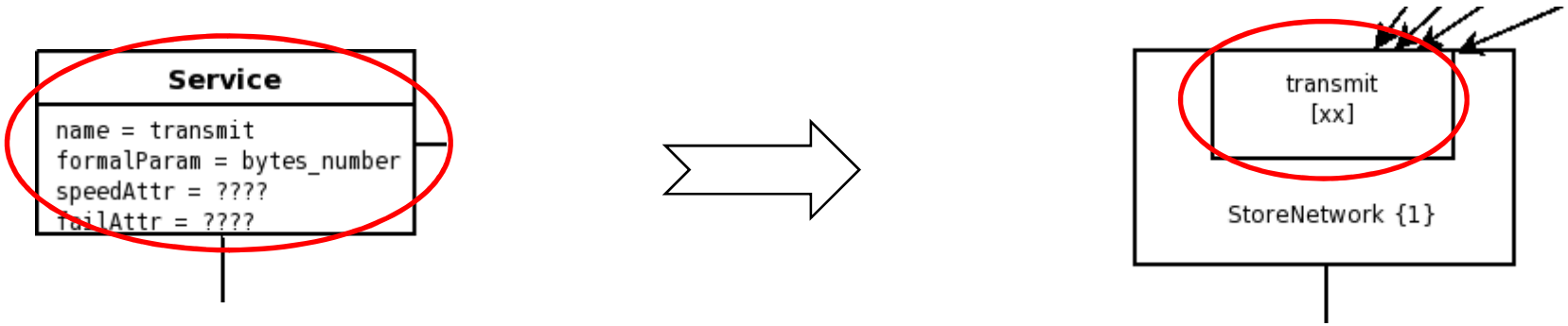
Transformation rules from KLAPER to LQN

- …and each KLAPER Resource that models an hardware device originates an LQN Processor.



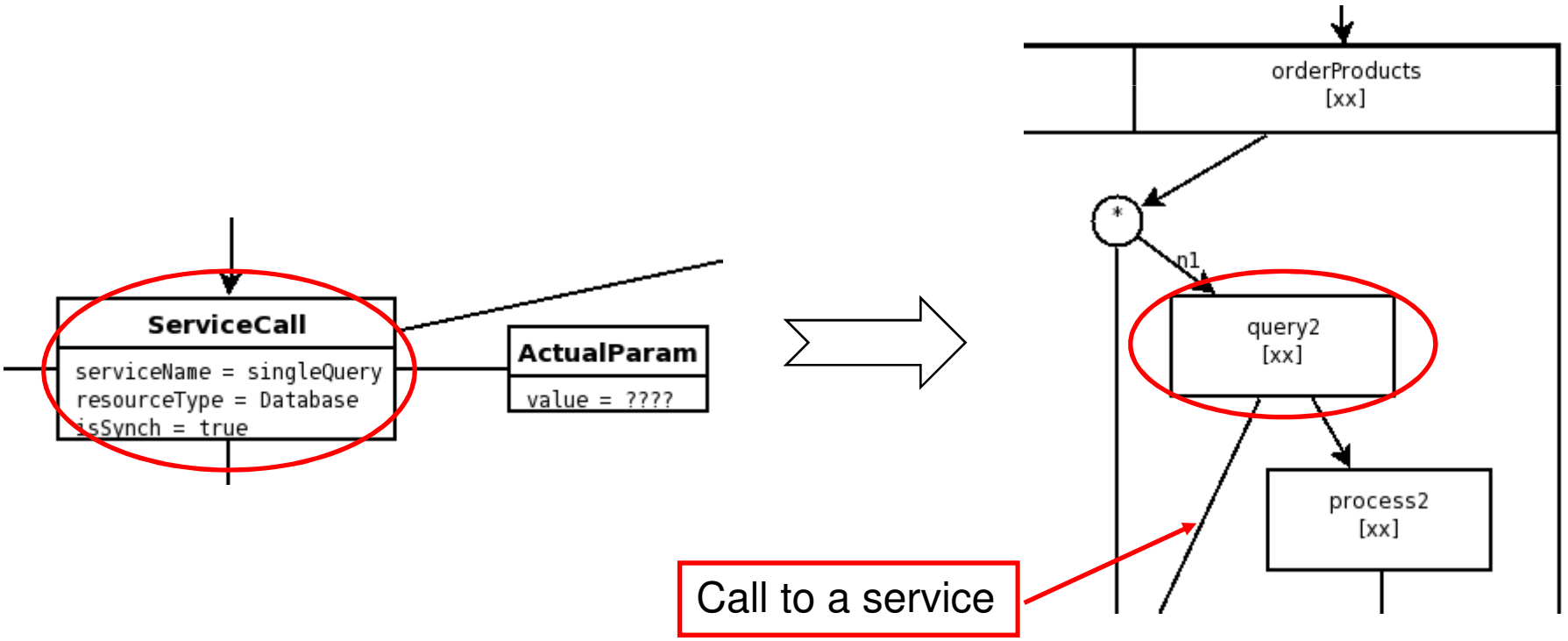
Transformation rules from KLAPER to LQN

- Each KLAPER Service becomes an Entry of a LQN Task.



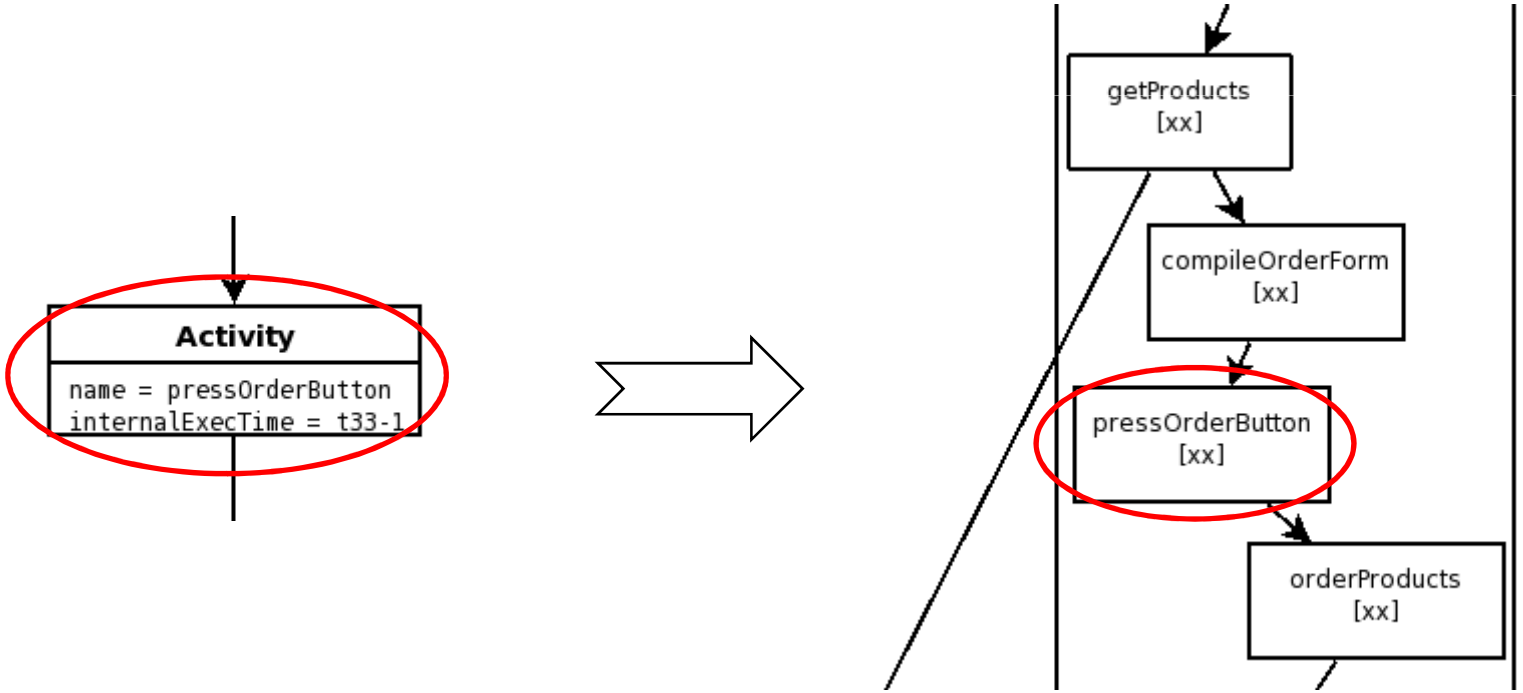
Transformation rules from KLAPER to LQN

- Each KLAPER ServiceCall is mapped onto an Activity (making a “call”) of an LQN Entry.



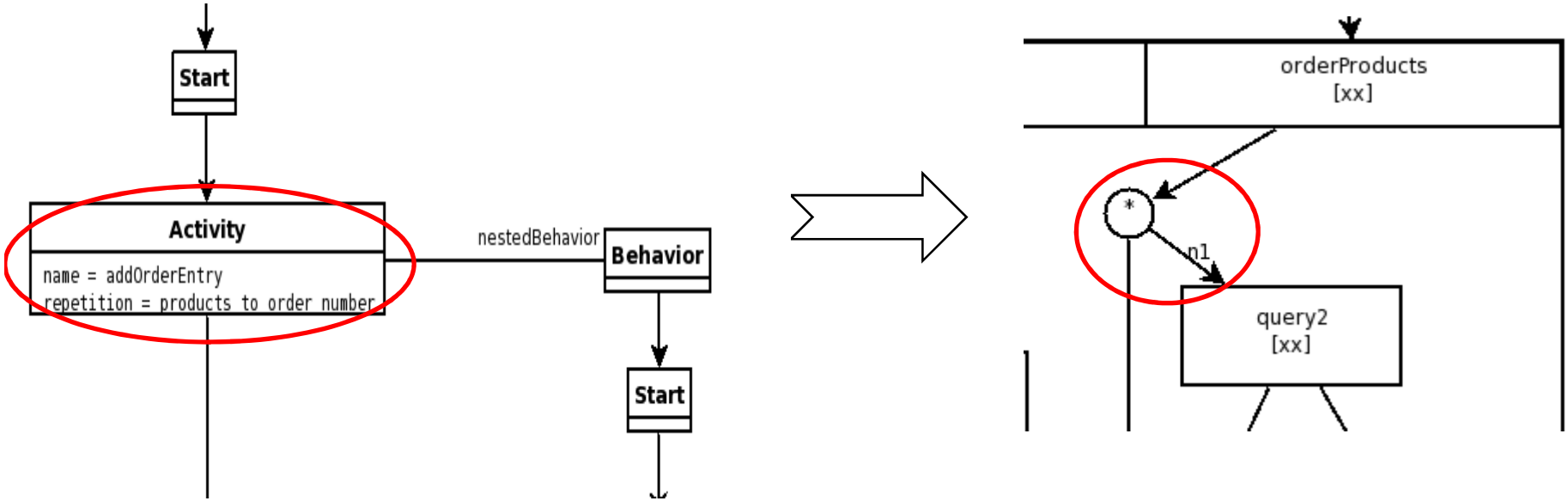
Transformation rules from KLAPER to LQN

- Each KLAPER Activity is mapped into an LQN Activity.

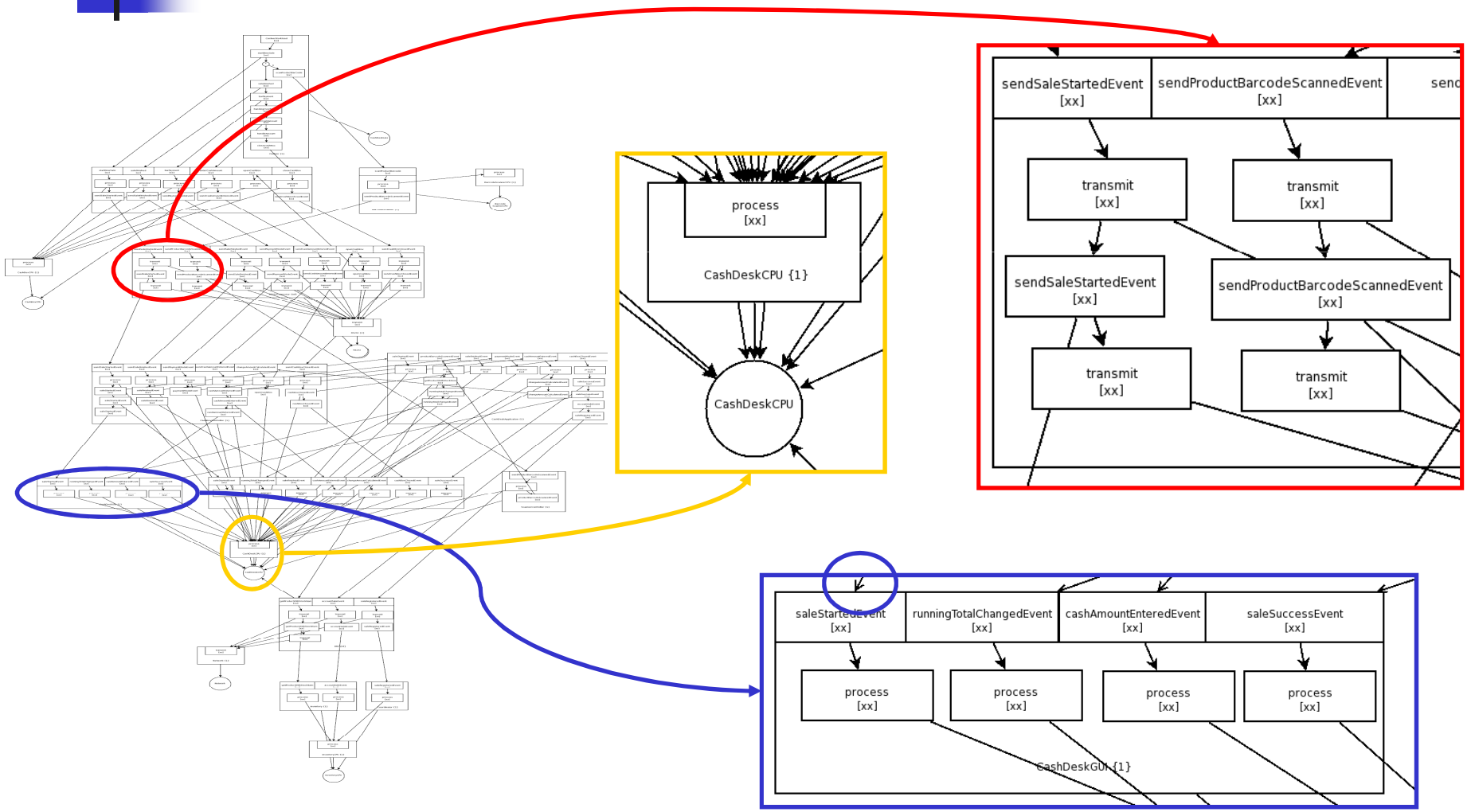


Transformation rules from KLAPER to LQN

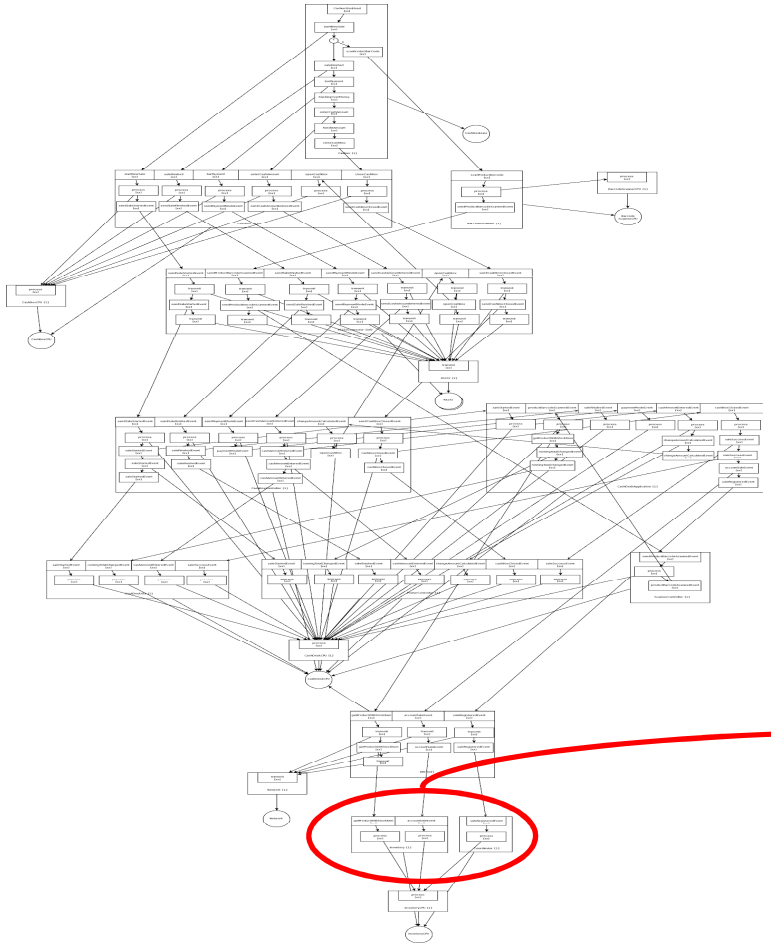
- Each KLAPER condition (or, and, loop) is directly mapped onto the corresponding LQN condition.

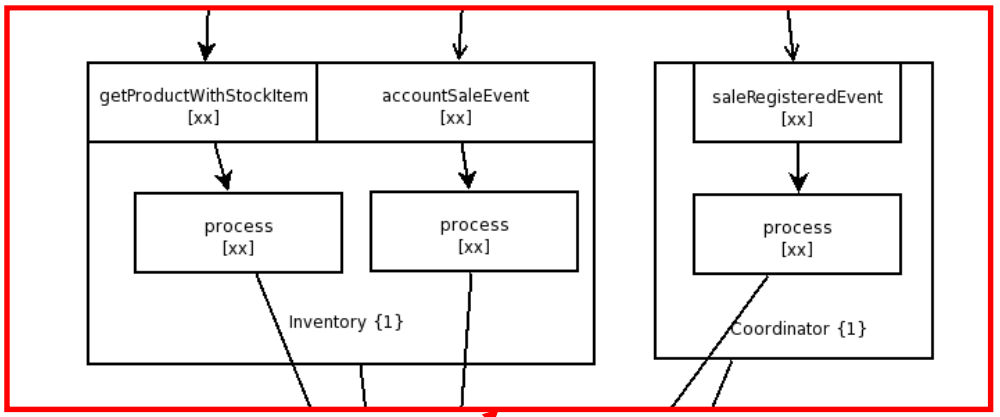


The LQN model for the CoCoME UC1 use case

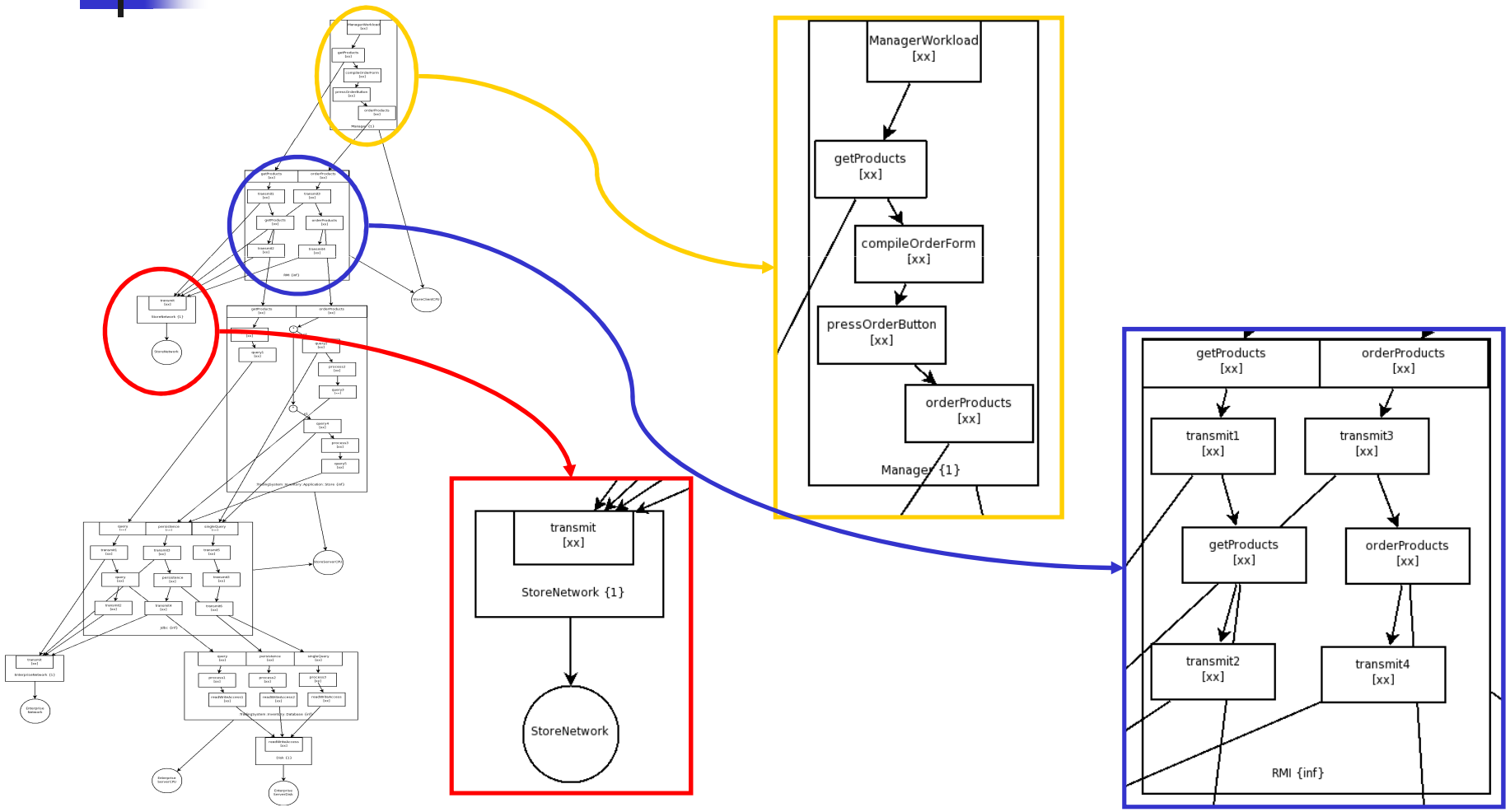


The LQN model for the CoCoME UC1 use case

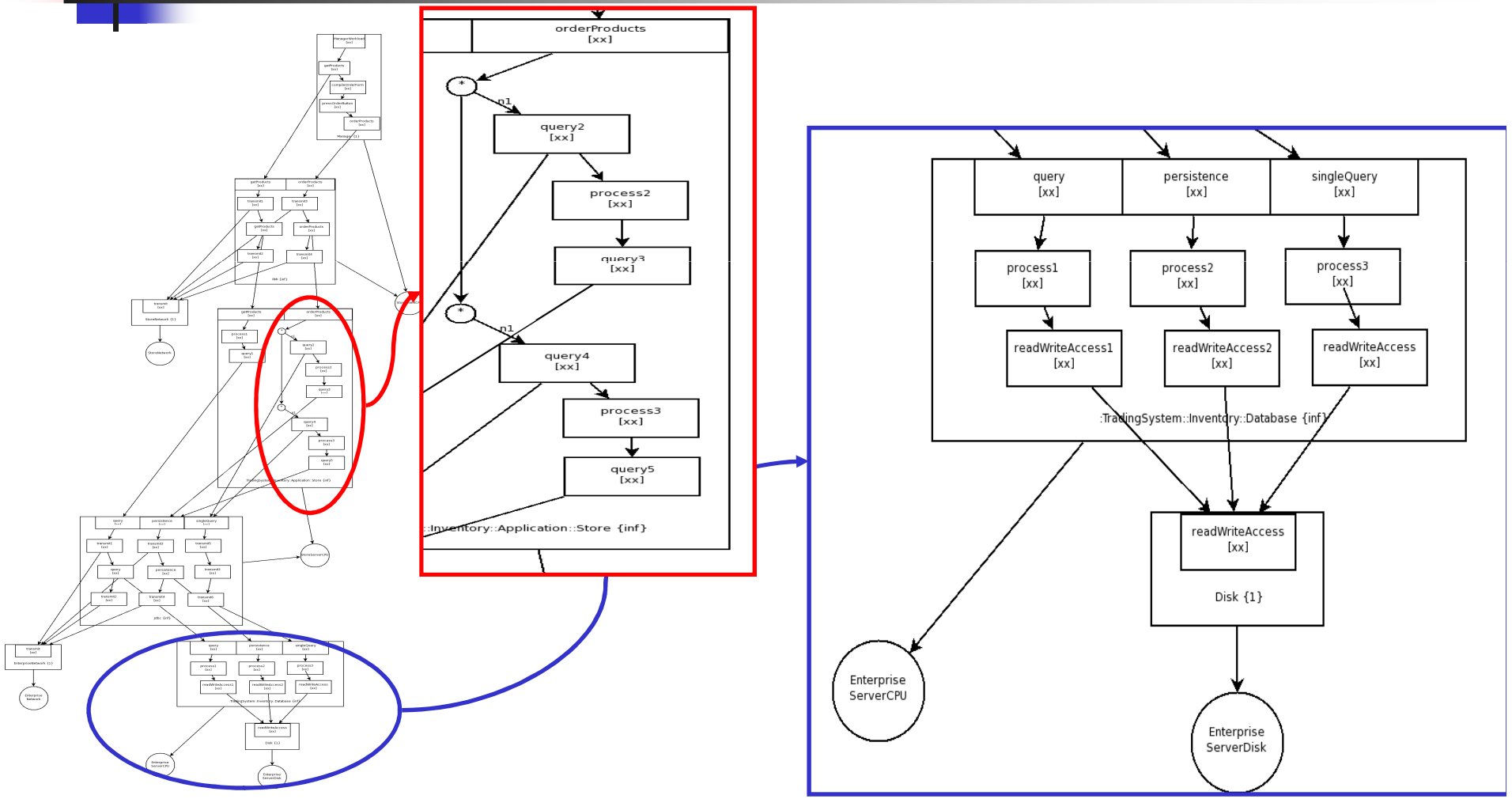


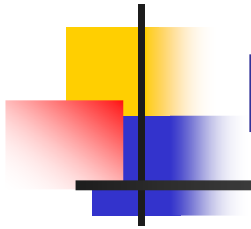


The LQN model for the CoCoME UC3 use case



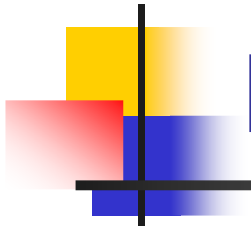
The LQN model for the CoCoME UC3 use case





Results of the analysis

- 4 different simulation configurations
- Base configuration:
  - Processors: Pentium III at 500 Mhz (1354 MIPS).
  - Networks: bandwidth of 100 Mbps
  - Database disk: average rate of 300 MBps

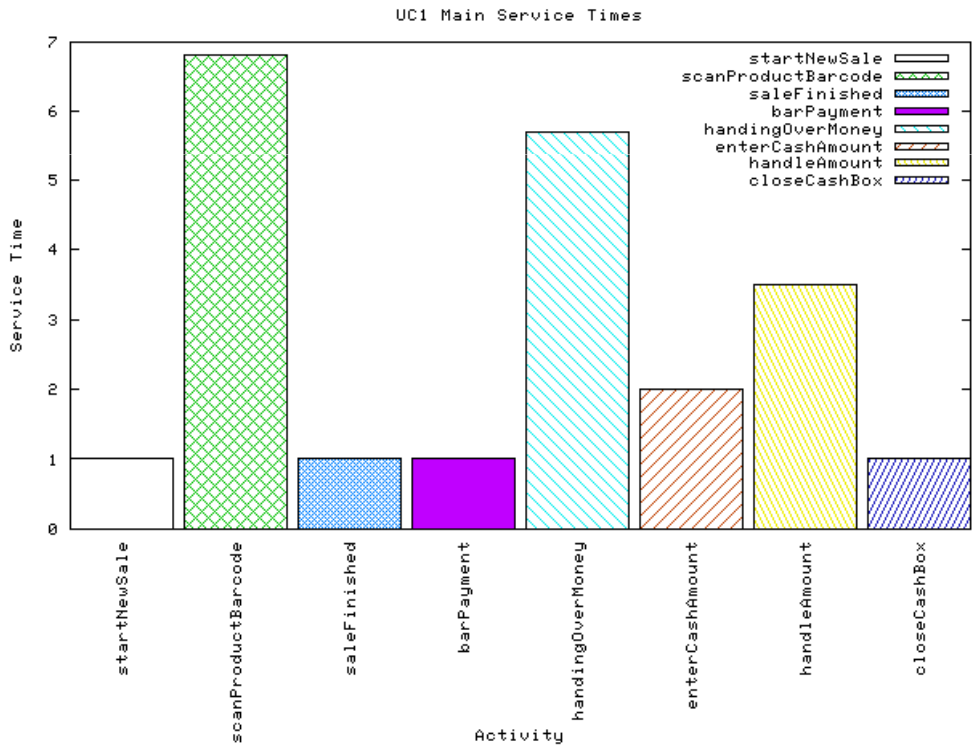


Results of the analysis

other configurations :

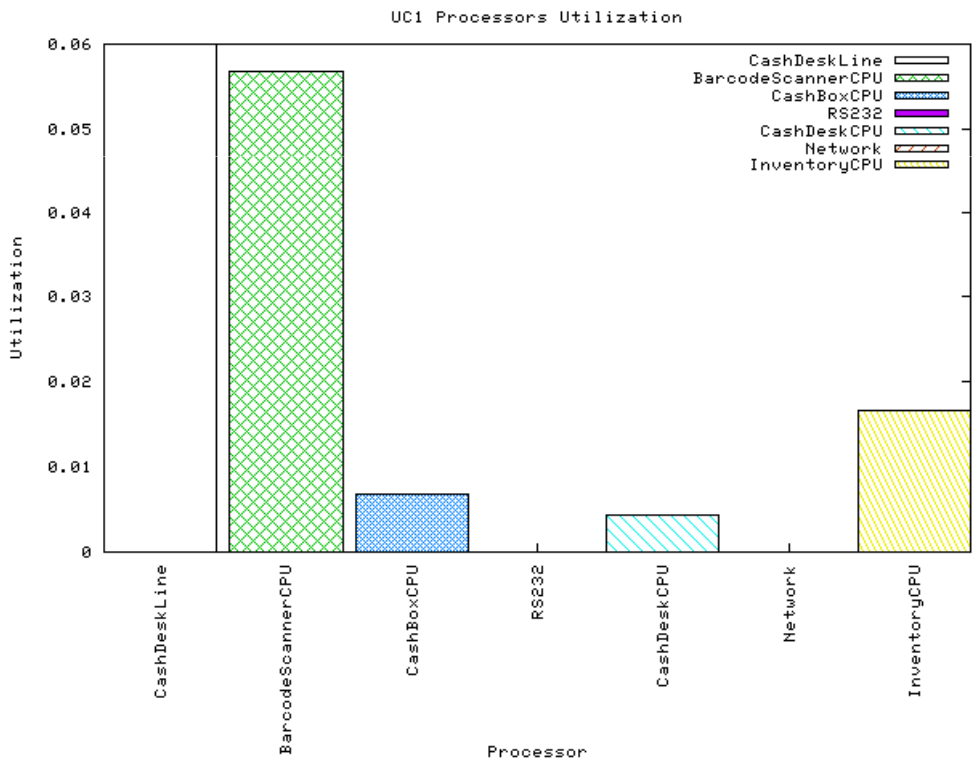
- “Cpu 2x” configuration:
  - Processors: double frequence.
- “Internet” configuration:
  - Network: bandwith of 4 Mbps (was 100 Mbps).
- “Slow disk” configuration:
  - Database disk: average rate of 100 MBps (was 300 MBps).

Results of the analysis



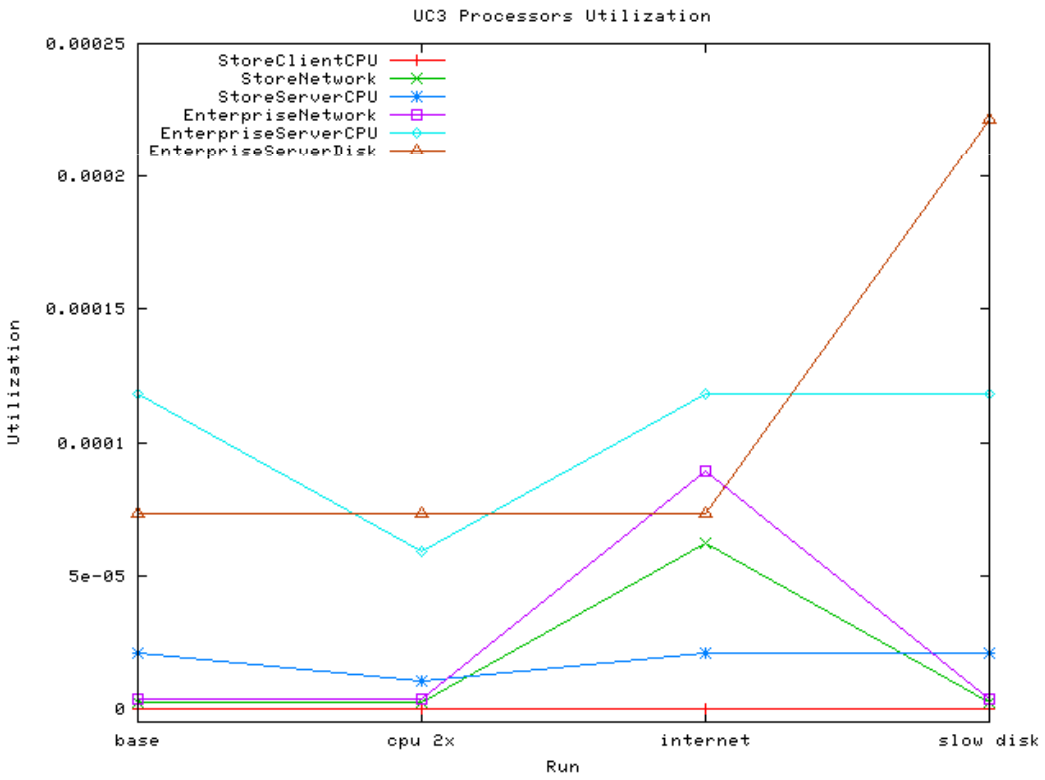
- UC1 is characterized by a lot of human interactions.
- Service times registered at the main workflow level
- A lot of time is spent in human actions (*scanProductBarcode*, *handingOverMoney*, *handleAmount*)
- The most expensive activity is *scanProductBarcode*, which is also repeated many times.

Results of the analysis



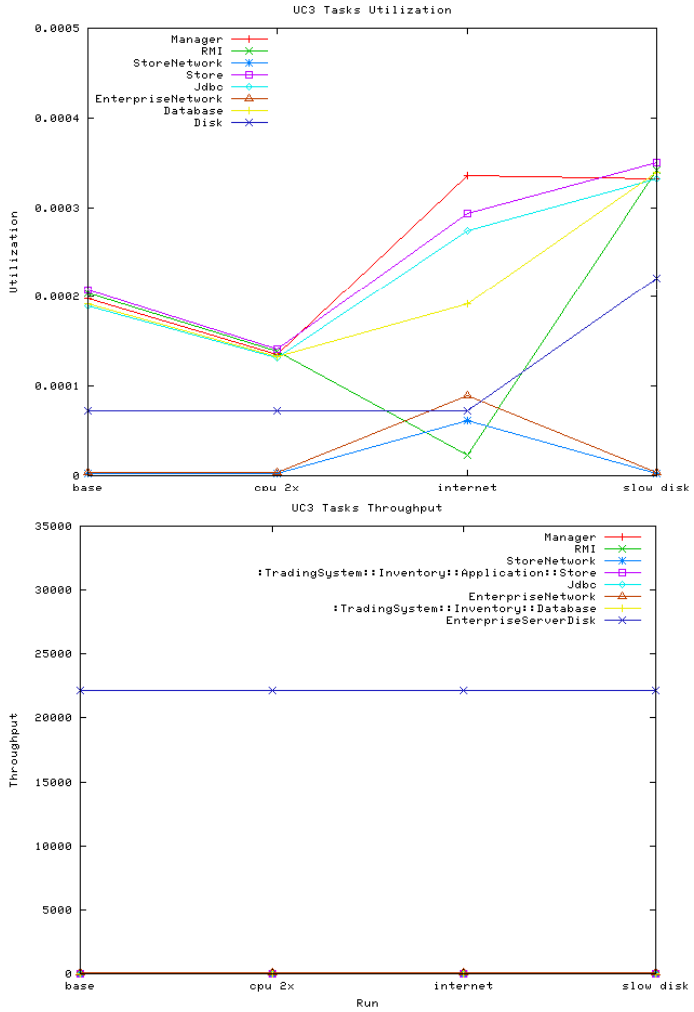
- Very low utilization values : the arrival rate of customers to the store is very small compared to the capacity of the system.
- Very low utilization of Network and RS232.
- *CashDeskLine* and *BarcodeScannerCPU* are the most used processors (the last one is where the *scanProductBarCode* service runs, see previous slide).

Results of the analysis



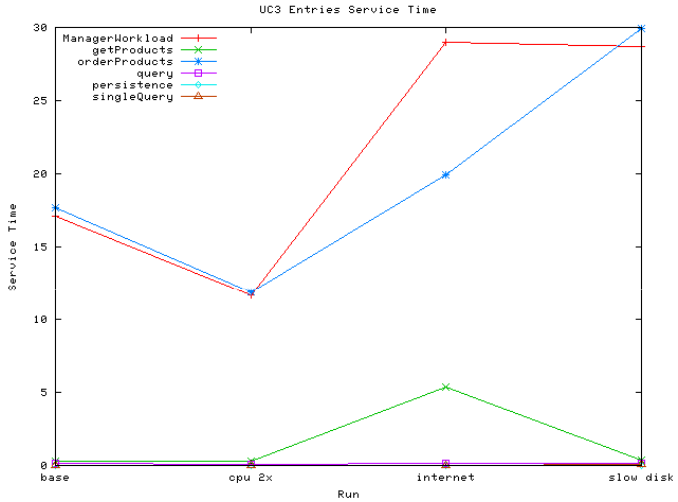
- UC3 is characterized by a smaller human interaction than UC1
- Cpu 2x: *EnterpriseServerCPU* and *StoreServerCPU* have a lower utilization due to the increased cpu frequency.
- Internet: less bandwith means higher time for transmission and therefore an increse of the utilization.
- Slow disk: all unchanged except for the disk utilization, which is increased due to the reduced transfer rate.

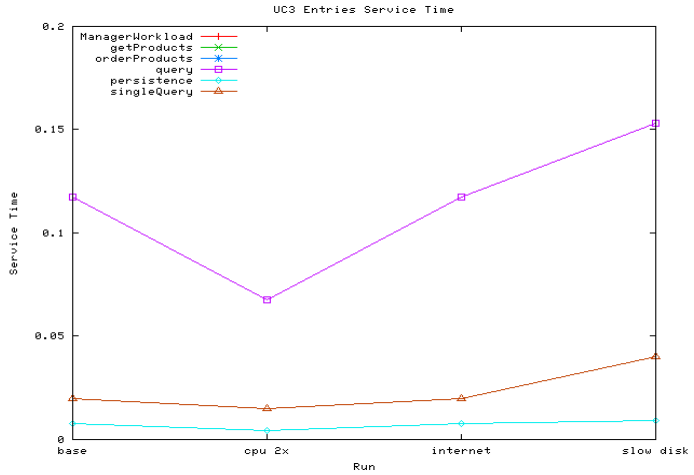
Results of the analysis



- Cpu 2x: a more powerful cpu means best performance in all the tasks (except for the disk one).
- Internet: all tasks need more time to execute except for RMI that receives less service requests for second (because we spent more time on the other tasks).
- Slow disk: network tasks are unchanged but all the other tasks have an higher utilization due to the fact that the disk in UC3 is used in every action.
- Throughput is the same for all the cases because the system load is very low.

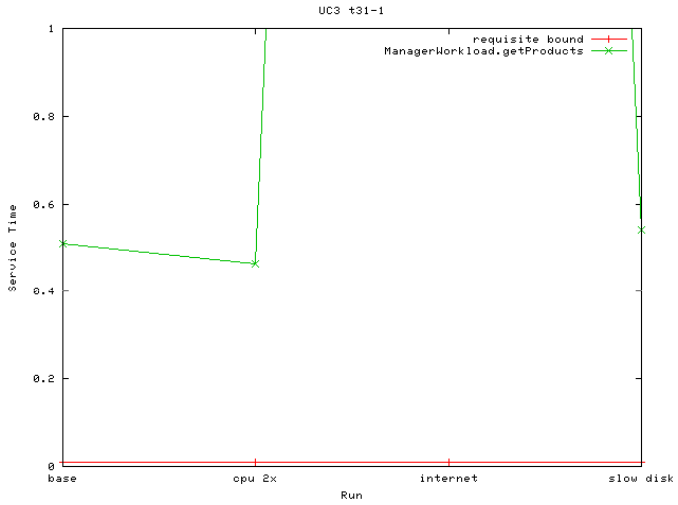
Results of the analysis

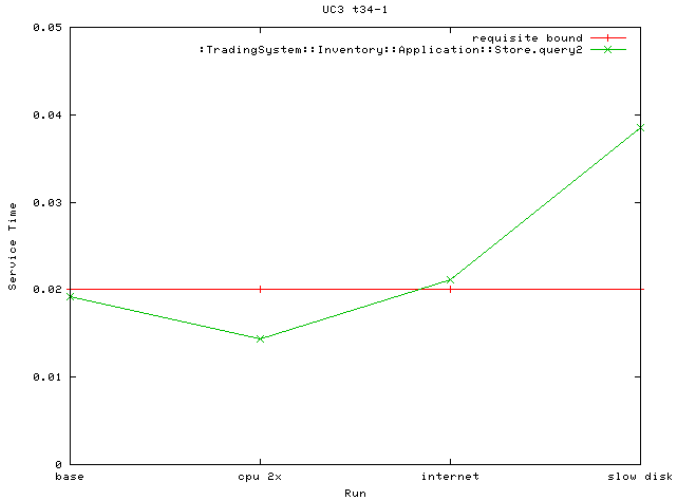




- Cpu 2x: more powerful cpu's mean less execution time.
- Internet: more time for transmission means higher service times (see the *ManagerWorkload* Entry).
- Slow disk: a lower rate means more time for each read.
- All as expected!

Results of the analysis

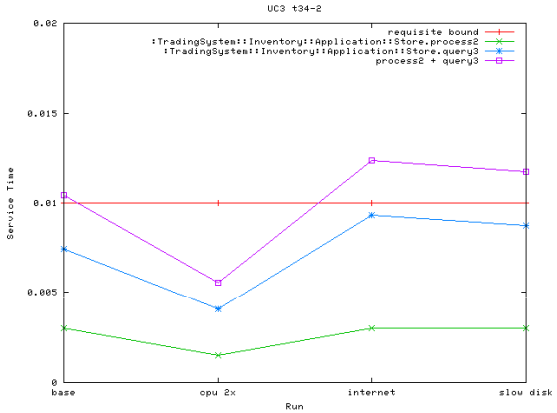


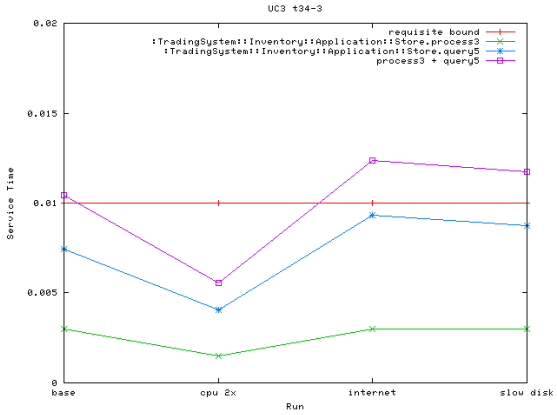


- Green line: simulated value.
- Red line: requisite bound.

- T31-1: time until showing the lists of all products and missing products.
- T34-1: time for queryng the inventory data store

Results of the analysis

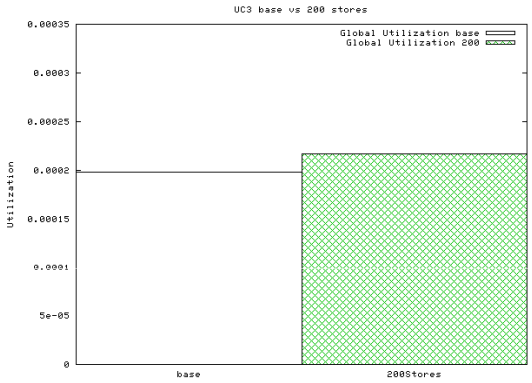


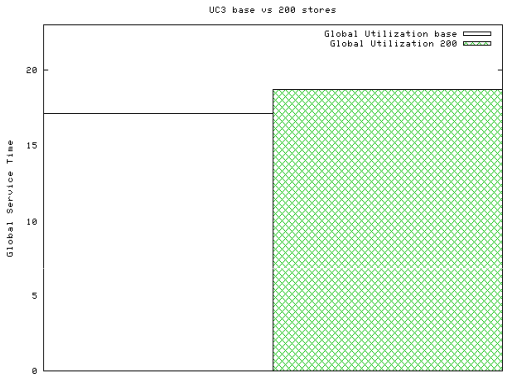


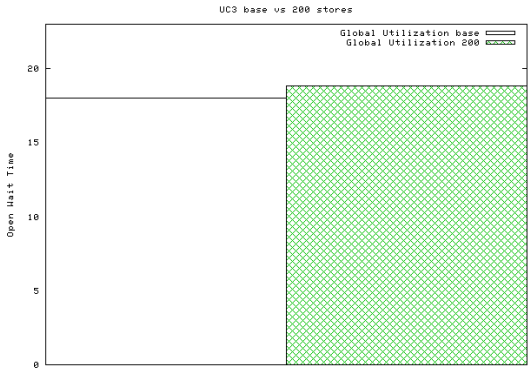
- Fuchsia line: simulated value.
- Red line: requisite bound.

- t34-2: time for creating a new order entry
- T34-3: time for creating a ne product order

Results of the analysis

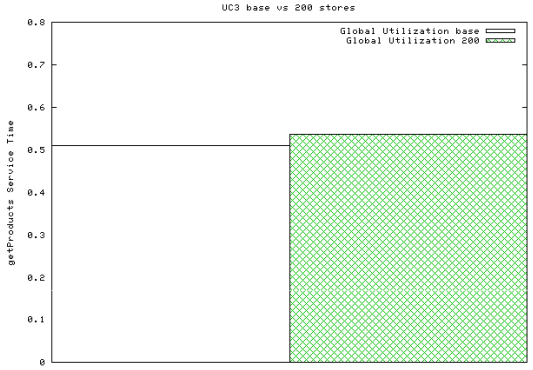


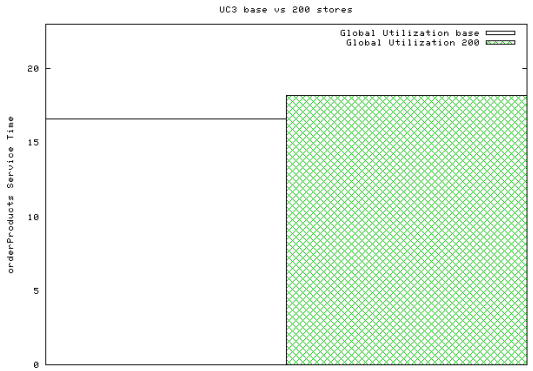




- 1 Store vs 200 Stores
- Simulated measures:
  - Utilization
  - Global Service Time
  - Entry Wait Time
  - Throughput

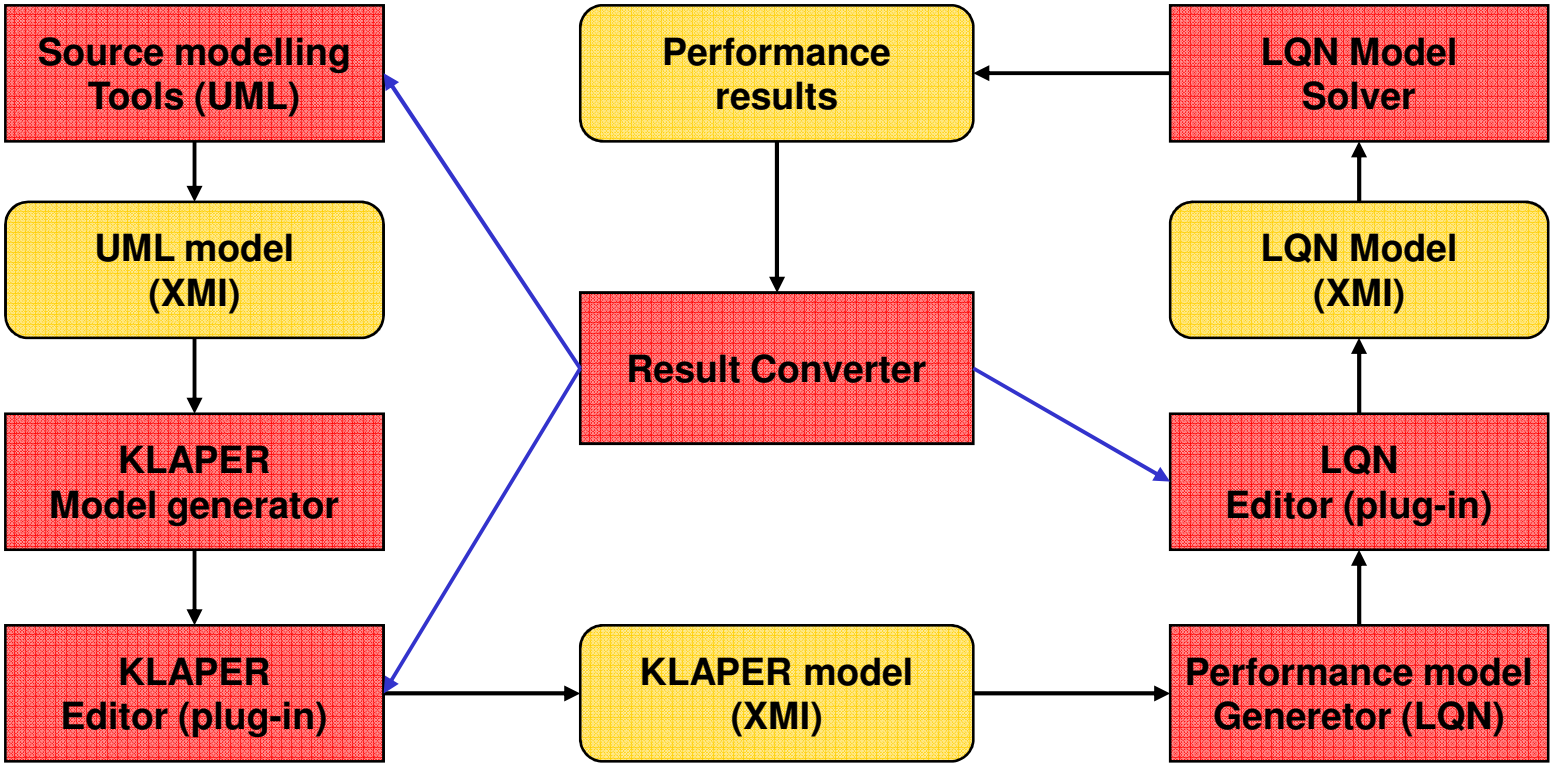
Results of the analysis





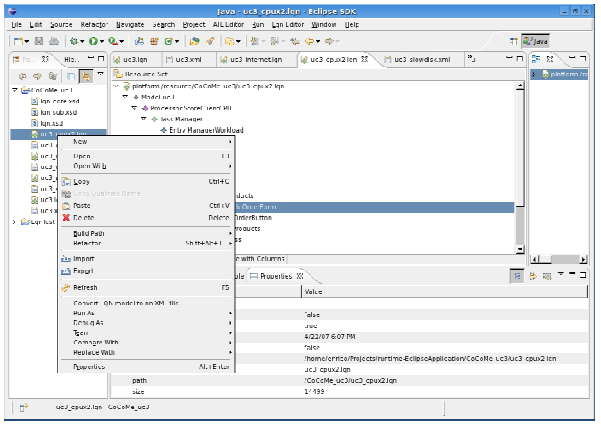
- 1 Store vs 200 Stores
- Simulated mesures:
  - getProducts service time
  - orderProducts service time

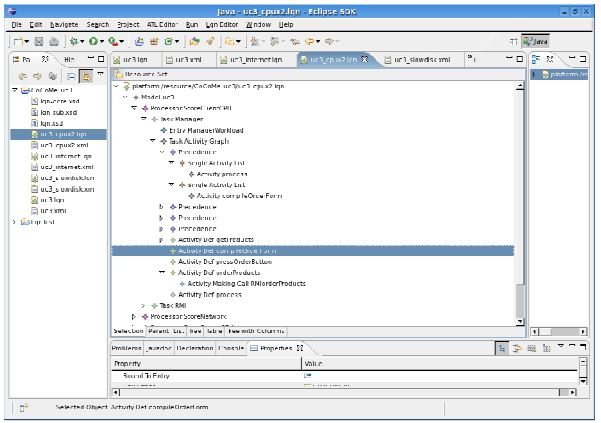
The KLAPER environment

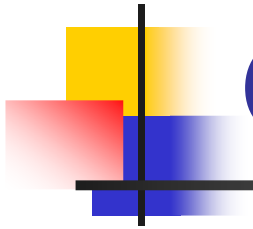


The KLAPER environment

- KLAPER tools developed on the EMF (Eclipse) platform:
  - KLAPER metamodel plugin
  - KLAPER editor plugin
  - LQN metamodel plugin
  - LQN editor plugin
  - KLAPER to LQN transformation plugin (work in progress)







Conclusions…

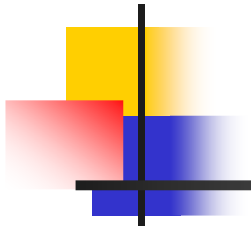
Need of automatic tools for the transformation from design models of component-based application to analysis models

A transformation framework centered around a kernel language called KLAPER

Captures the relevant information for the analysis of non-functional characteristics of component-based systems.
Facilitates (hopefully …) the transformation definition.

CoCoME: KLAPER has been used to derive a Layered Queueing Network, starting from the UML model annotated according to the SPT profile.

Definition of a (partially) automated environment



…Future works

The long-term goal of our research is:

to enhance the implementation of this framework,
(e.g., automatic model transformations using QVT-based languages)

to provide system designers with a richer toolkit that allows to generate automatically different performance and reliability models starting from design models.

For additional information see: http://klaper.sf.net