Including CORBA performance details into MDA System models

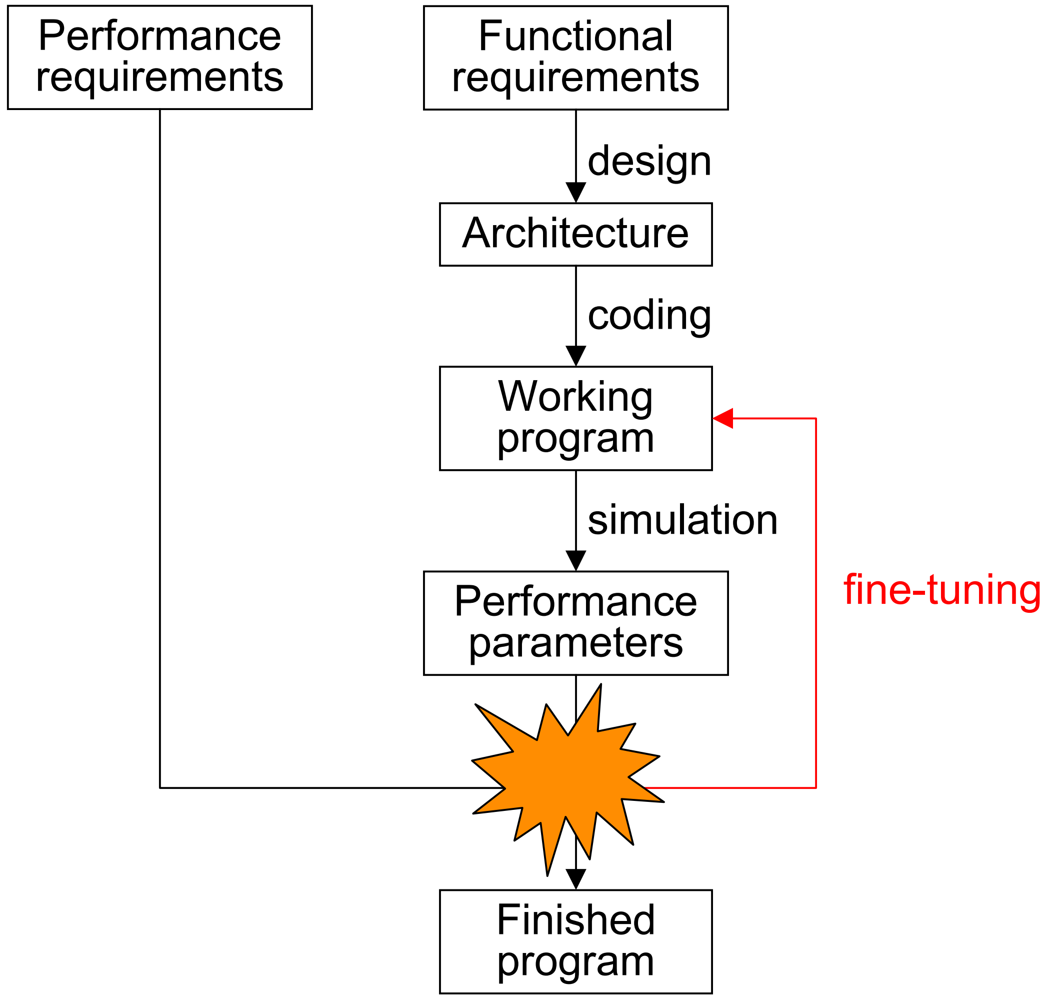
Tom Verdickt, Bart Dhoedt,
Frank Gielen, Piet Demeester

INTEC
UNIVERSITEIT GENT

Overview

- Software Performance Engineering
- SPE + MDA
- Middleware
- Transformation
- Conclusion

Traditional software design



Problem

- Traditional software design: functionality
- Performance only checked in finished product
- Result:
  - changes and fine-tuning in final stages
    - expensive (over budget)
    - time-consuming (missed deadlines)
  - poor performance in final system (bad software)
- Software quality ≈ performance

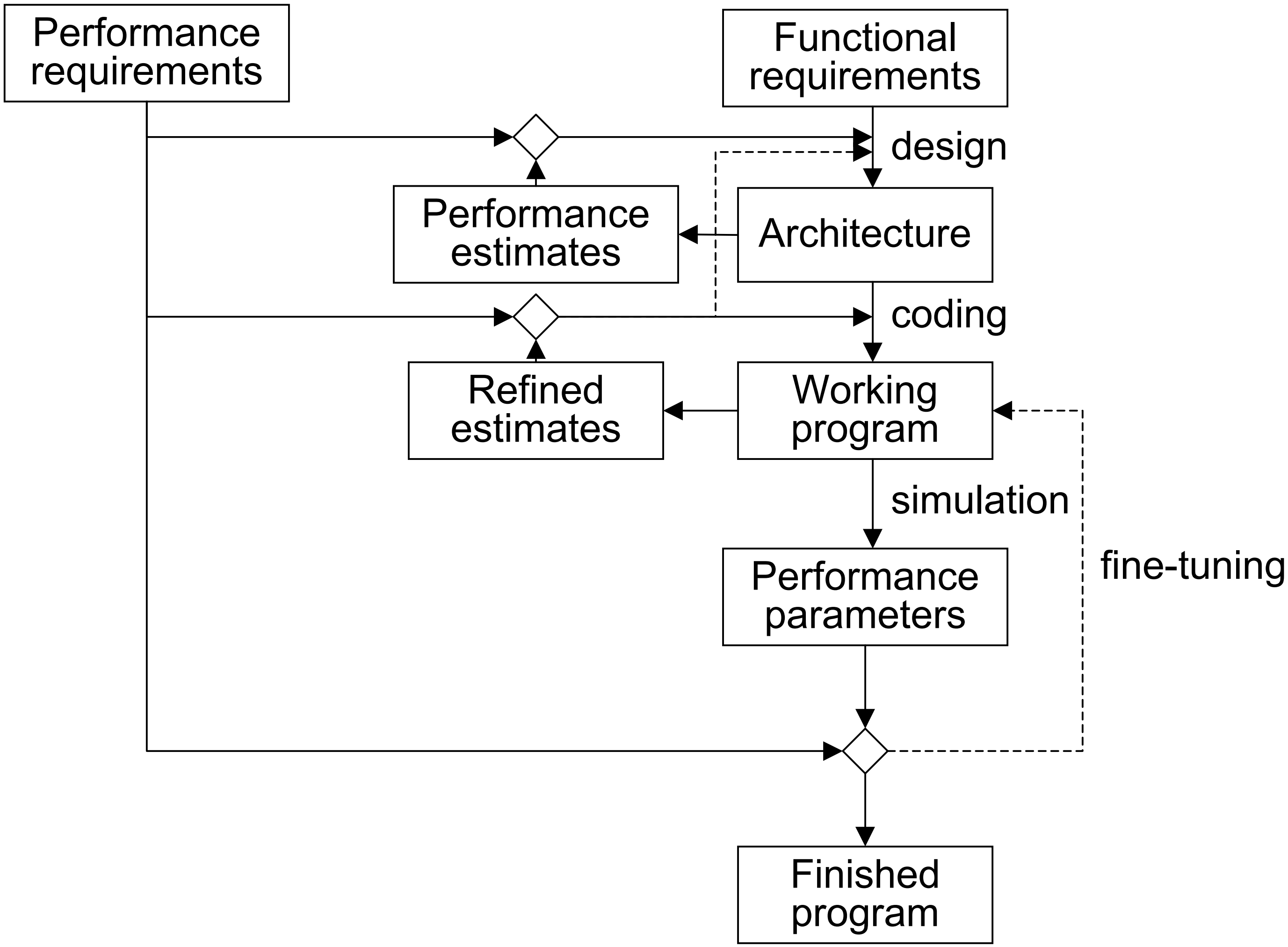
⇒many projects fail because of poor performance

  - 1/3 fails completely
  - 40% too late or over budget
  - many of those because of poor performance

Software Performance Engineering (1)

- Performance analysis during complete design, starting as early as possible
- Methods:
  - performance models (queueing networks, Petri nets)
  - quantitative methods (MVA) and simulations
- Build performance into design (don't try to add it later)
- Current status: accurate modeling of non-distributed systems

Software Performance Engineering (2)



SPE problems

- Performance models use dedicated modelling languages
  - Designers need to learn a new modelling language
  - Extra models needed => time-consuming
  - Synchronize changes
- General-purpose modelling languages lack performance features
- Solutions:
  - performance extensions (e.g. UML profiles)
  - automatic transformations

Overview

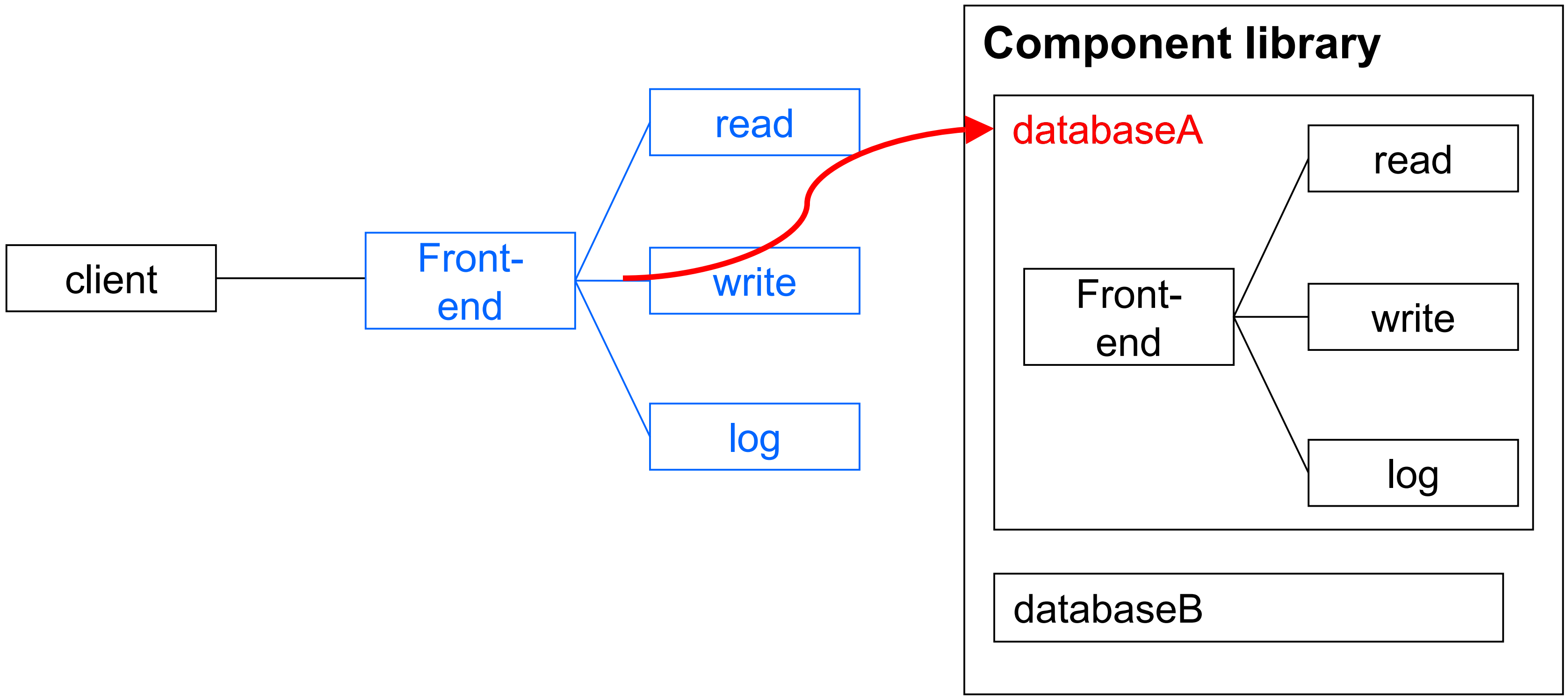
- Software Performance Engineering
- SPE + MDA
- Middleware
- Transformation
- Conclusion

SPE + MDA

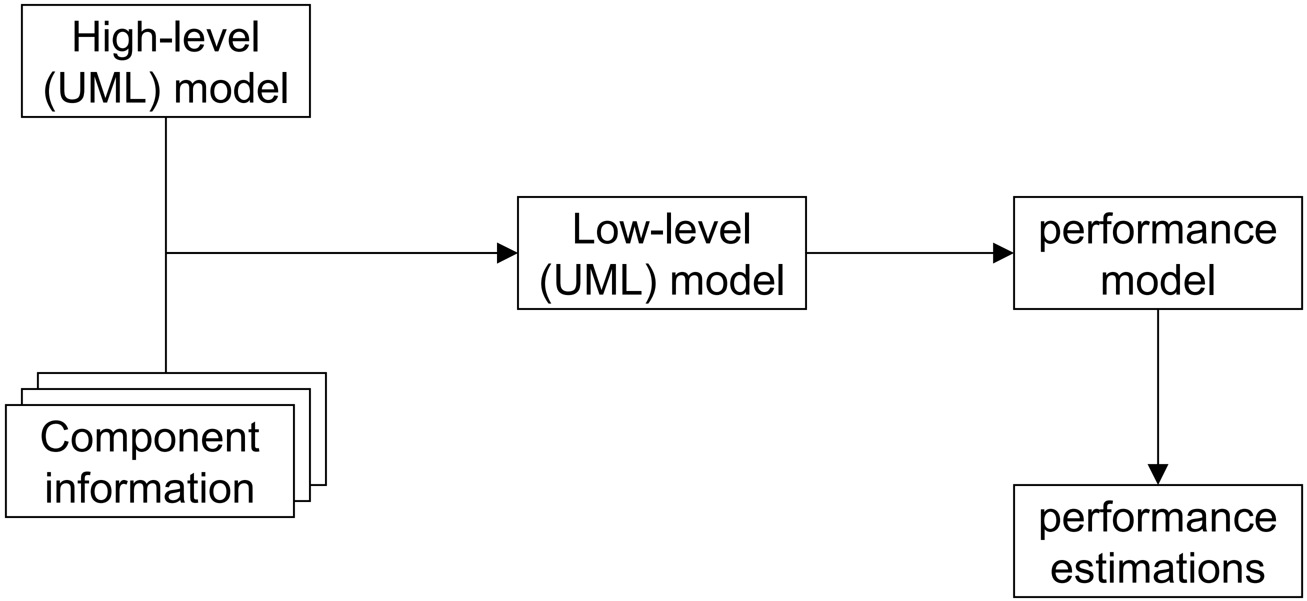
- Many performance details needed
  - platform
  - middleware
  - third-party components
  - etc.
- Goal: (semi-) automatic performance evaluation
- MDA/UML provides necessary features:
  - solid modelling formalism
  - different abstraction levels
  - include component models
  - model refinement

PIM-to-PSM transformation

Ideal performance modelling process



Ideal performance modelling process

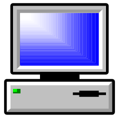


Overview

- Software Performance Engineering
- SPE + MDA
- Middleware
- Transformation
- Conclusion

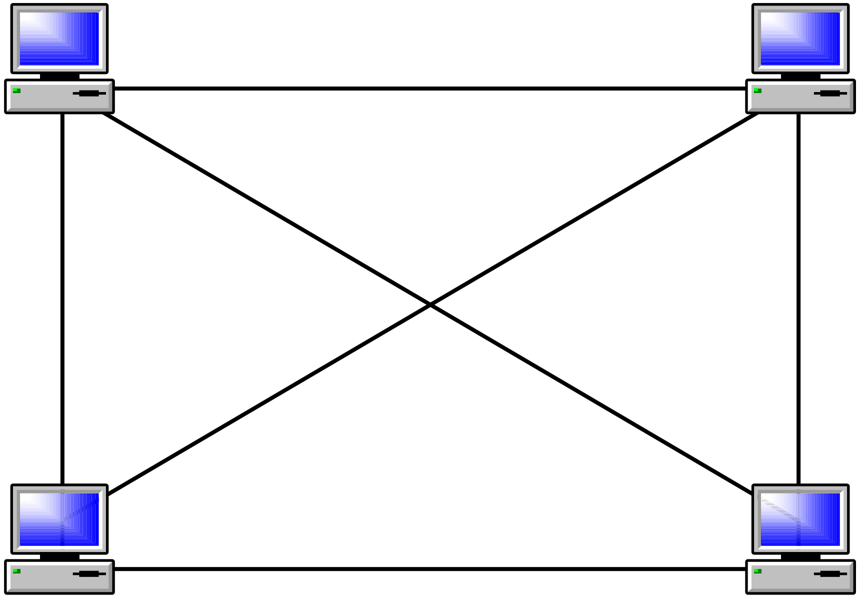
Growing system complexity

- Single computer



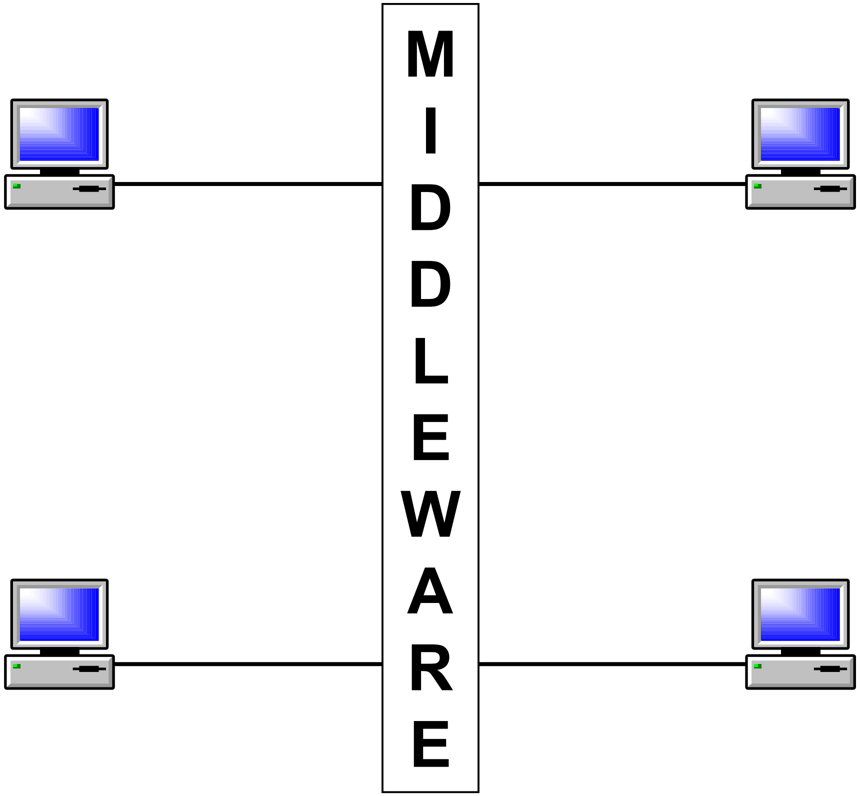
Growing system complexity

- Single computer
- Distributed system



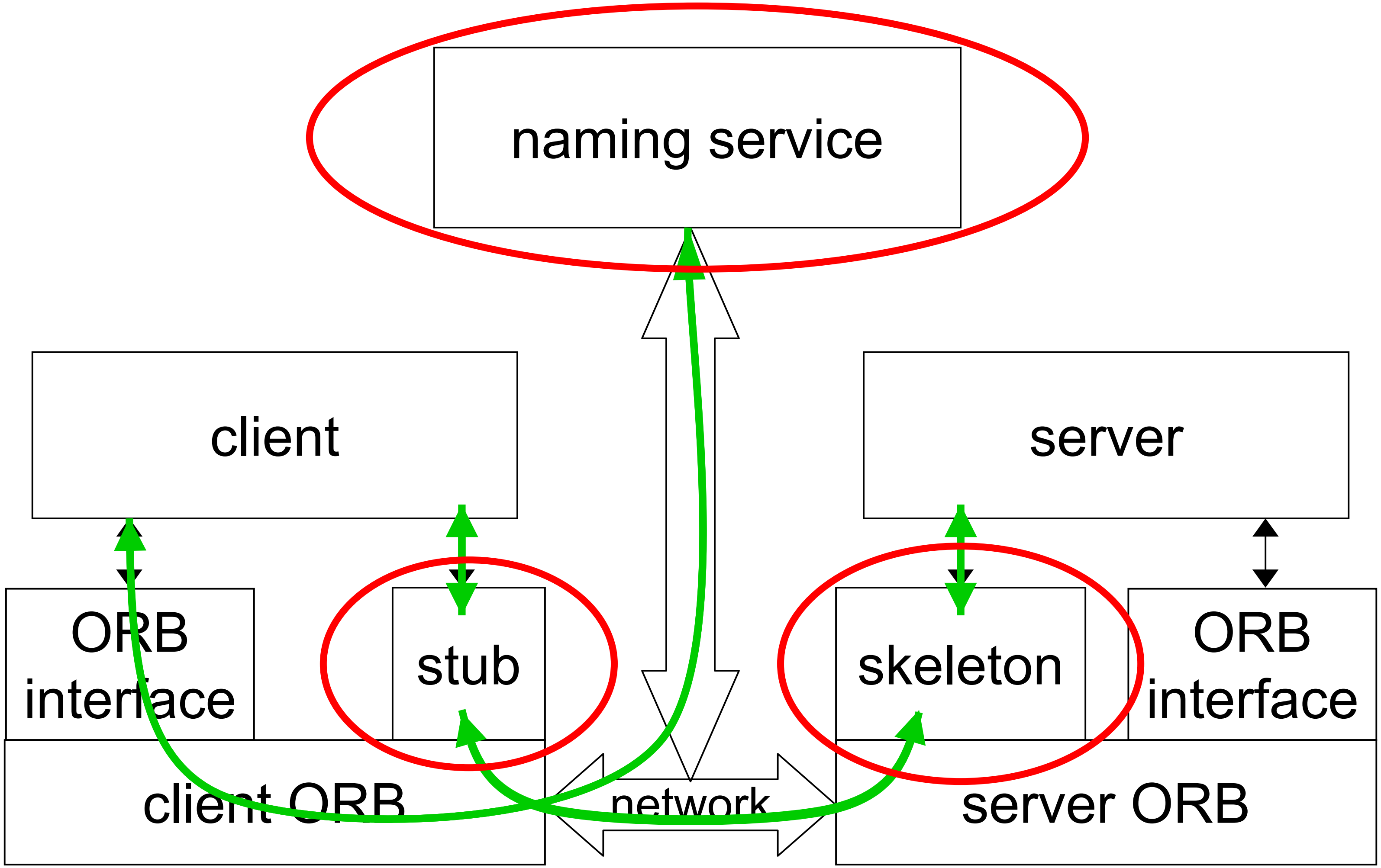
Growing system complexity

- Single computer
- Distributed system
- Distributed system using middleware



Middleware

- Goal: provide interoperability between various components of a distributed system
- Examples: CORBA, JINI, RMI
- CORBA benefits:
  - independence from programming language, architecture, platform
  - event handling
  - location transparency (naming service)


naming service
client
server
ORB
interface
stub
skeleton
ORB
interface
client ORB
network
server ORB

Overview

- Software Performance Engineering
- SPE + MDA
- Middleware
- Transformation
- Conclusion

Input

- High-level PSM, using UML
  - collaboration diagram
  - deployment diagram
  - activity diagram
- Description of middleware details
  - type (e.g. CORBA)
  - performance information
  - deployment
- Naming convention to link the different models together
- A library of detailed middleware models (can be included implicitly in the transformation algorithm)

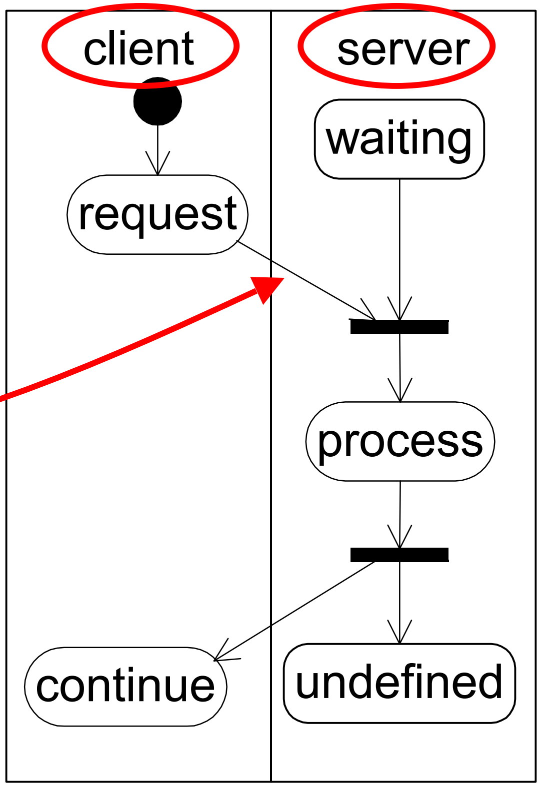
Output

- Low-level PSM, using UML
  - several collaboration diagrams
  - deployment diagram
  - activity diagram
- Diagrams contain performance information using the *UML Profile for schedulability, performance and time*

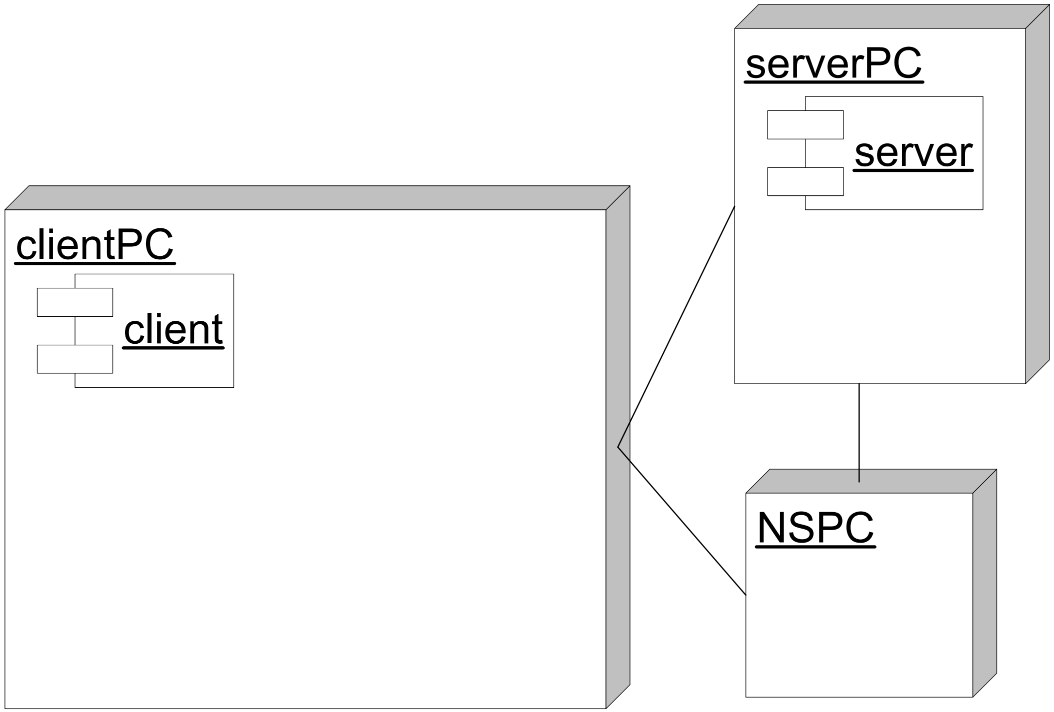
Transformation: component names

```
<middleware>
    <link id="link1" type="CORBA" NSref="NS1">
        <call cref="G.9" />
    </link>

    <NS id="NS1" host="xmi.17" />
</middleware>
```



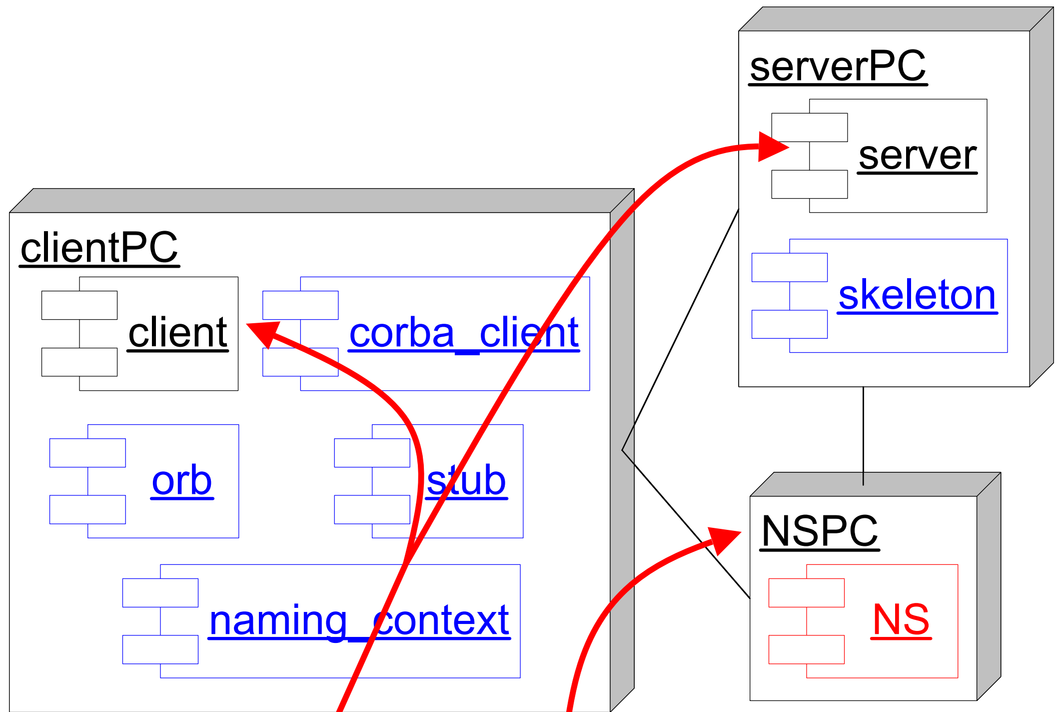
Transformation: deployment




```
<middleware>
    <link id="link1" type="CORBA" NSref="NS1">
        <call cref="G.9" />
    </link>

    <NS id="NS1" host="xmi.17" />
</middleware>
```

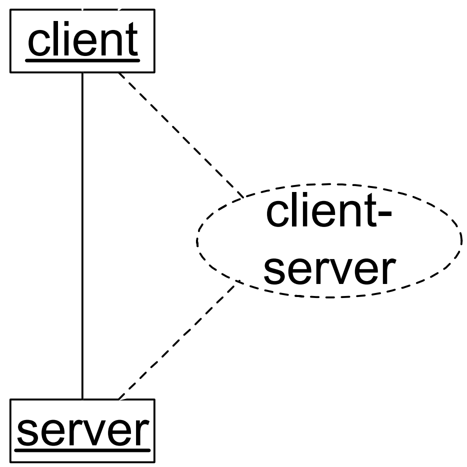
Transformation: deployment




```
<middleware>
    <link id="link1" type="CORBA" NSref="NS1">
        <call cref="G.9" />
    </link>

    <NS id="NS1" host="xmi.17" />
</middleware>
```

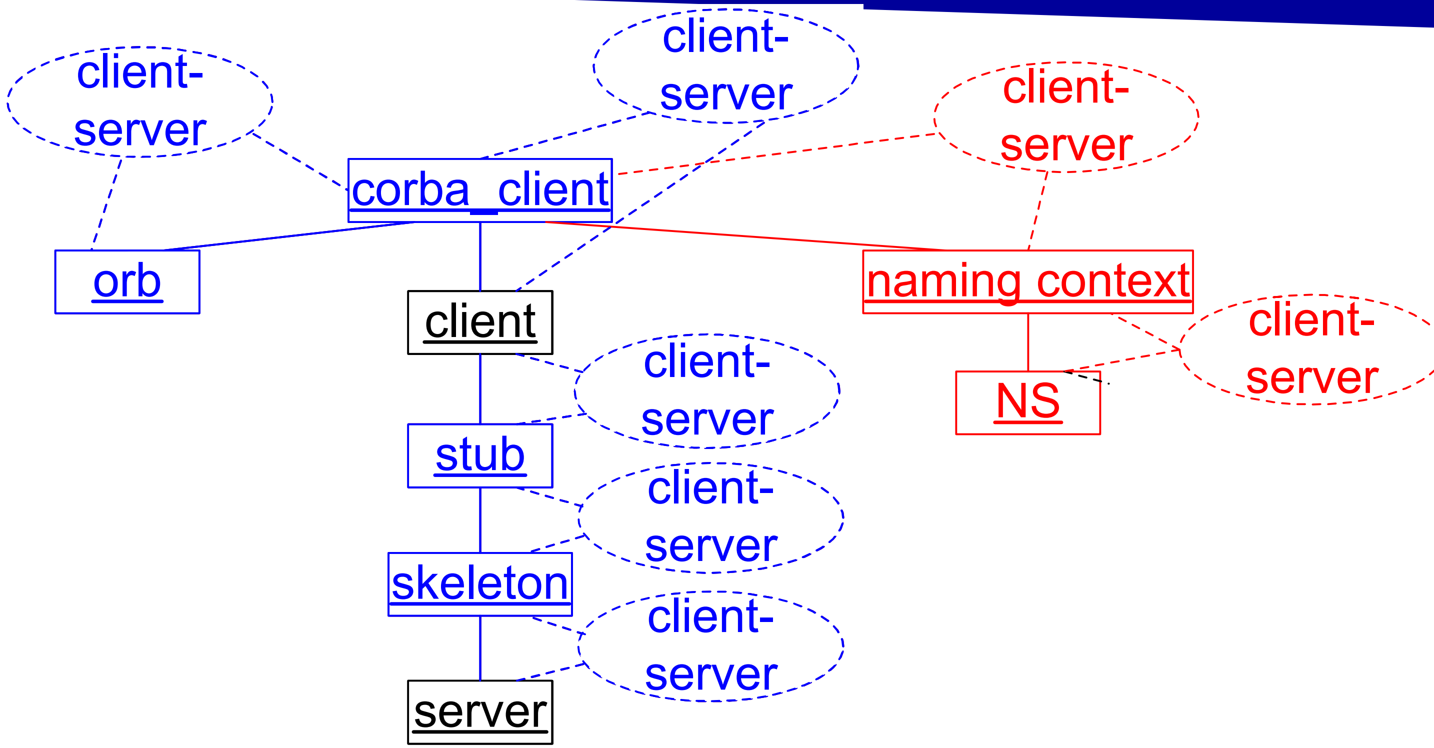
Transformation: collaboration




```
<middleware>
    <link id="link1" type="CORBA" NSref="NS1">
        <call cref="G.9" />
    </link>

    <NS id="NS1" host="xmi.17" />
</middleware>
```

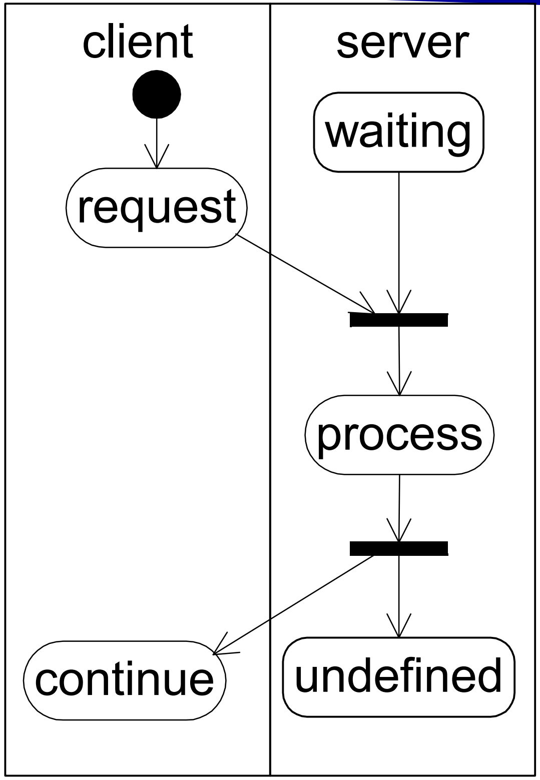
Transformation: collaboration




```
<middleware>
    <link id="link1" type="CORBA" NSref="NS1">
        <call cref="G.9" />
    </link>

    <NS id="NS1" host="xmi.17" />
</middleware>
```

Transformation: activity, input



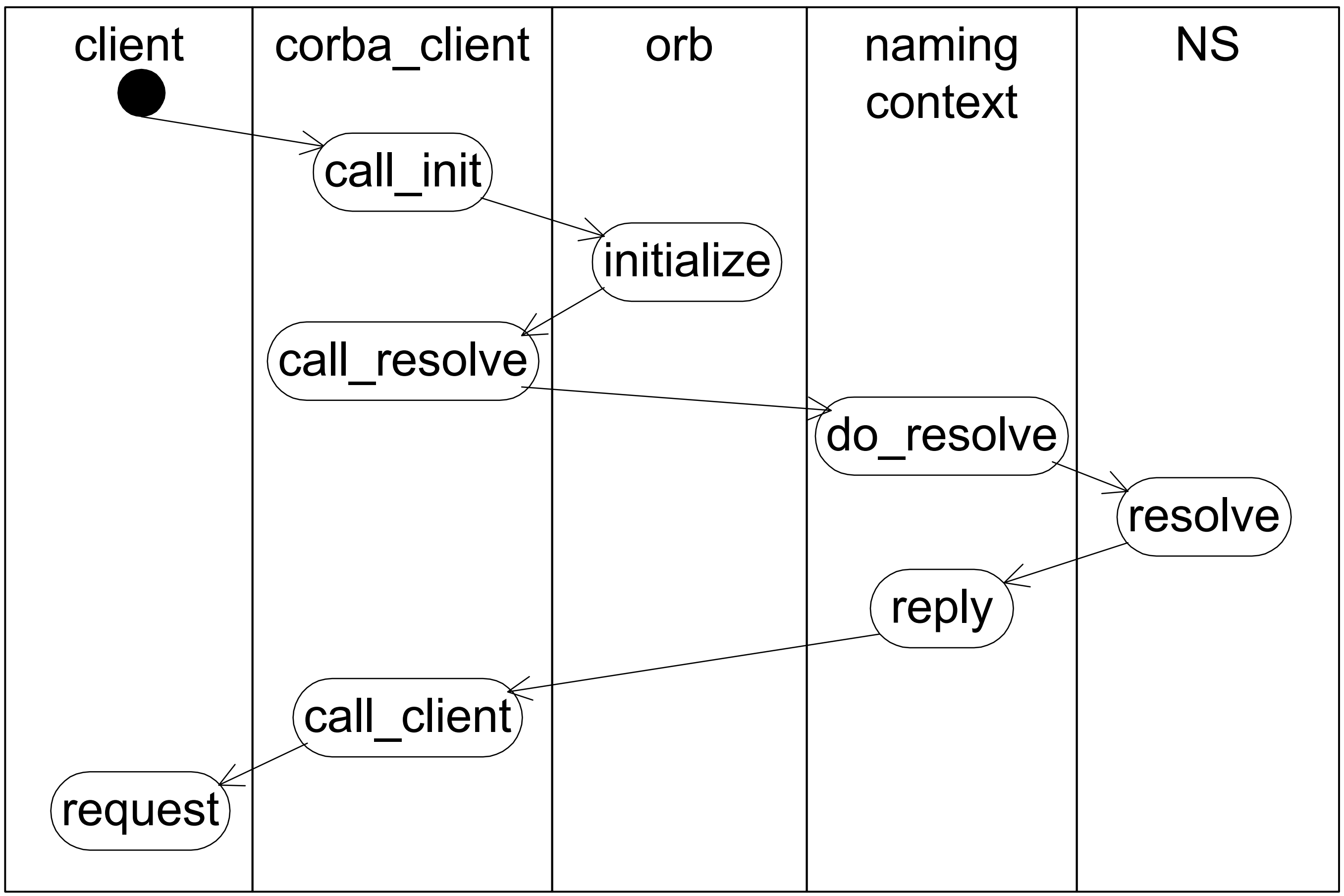

```
<middleware>
    <link id="link1" type="CORBA" NSref="NS1">
        <call cref="G.9" />
    </link>

    <NS id="NS1" host="xmi.17" />
</middleware>
```

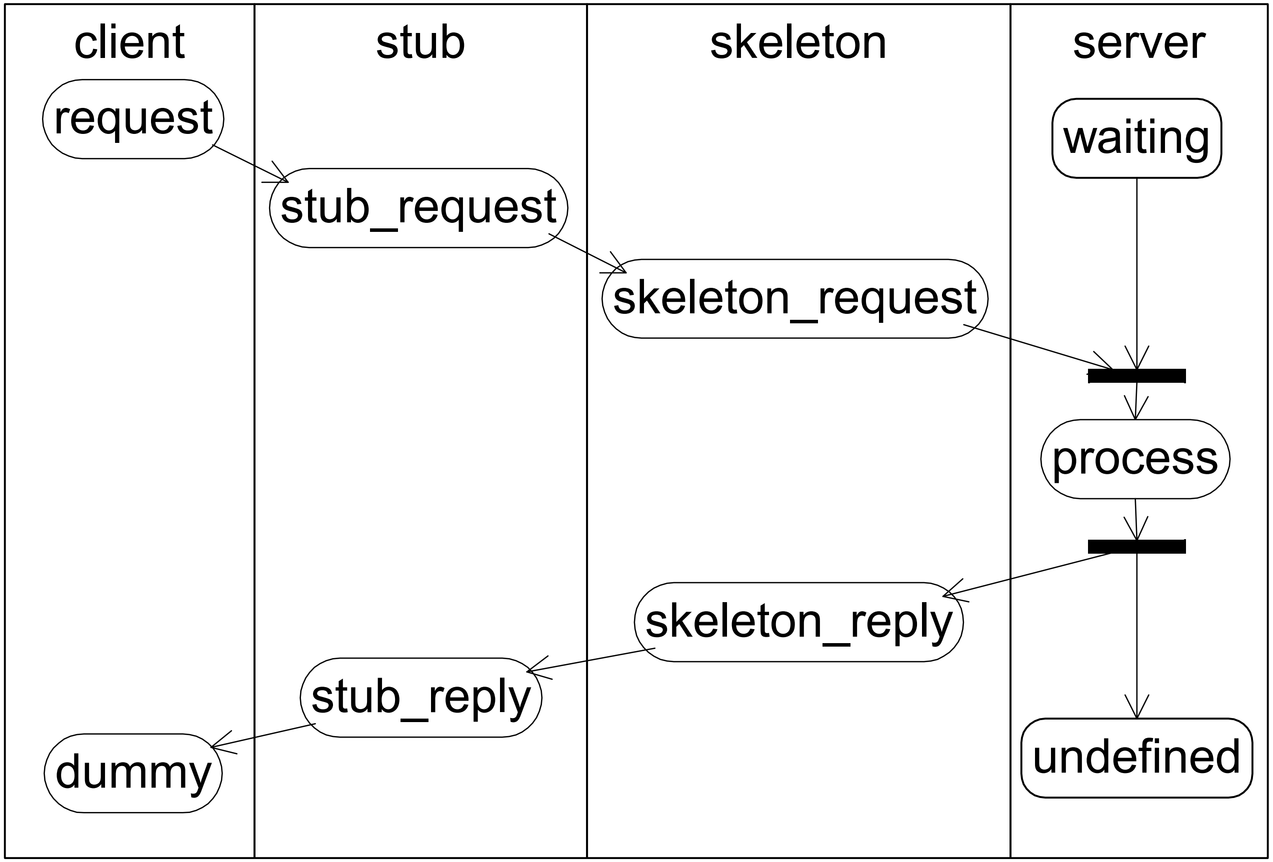
Transformation: activity

- initialization
- get server reference (Naming Service)
- call server
  - stub
  - skeleton
- destroy orb and clean up

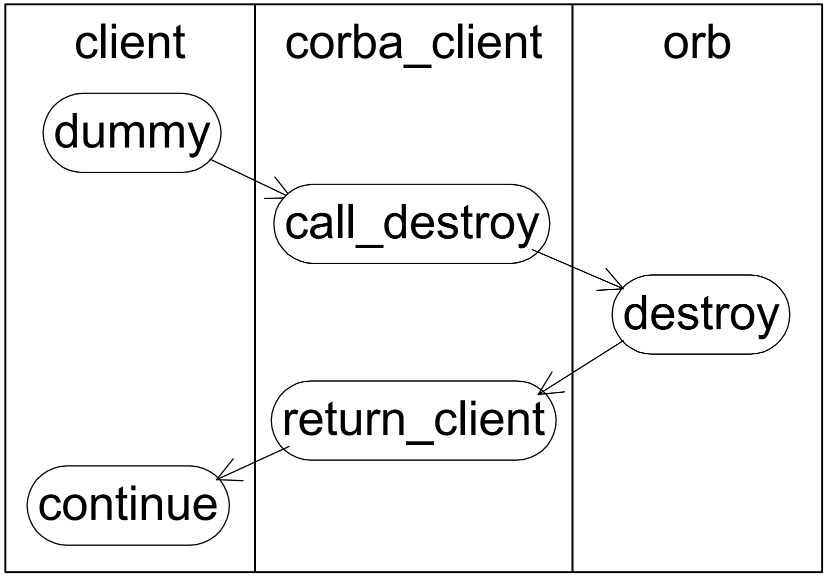
Transformation: activity, initialization



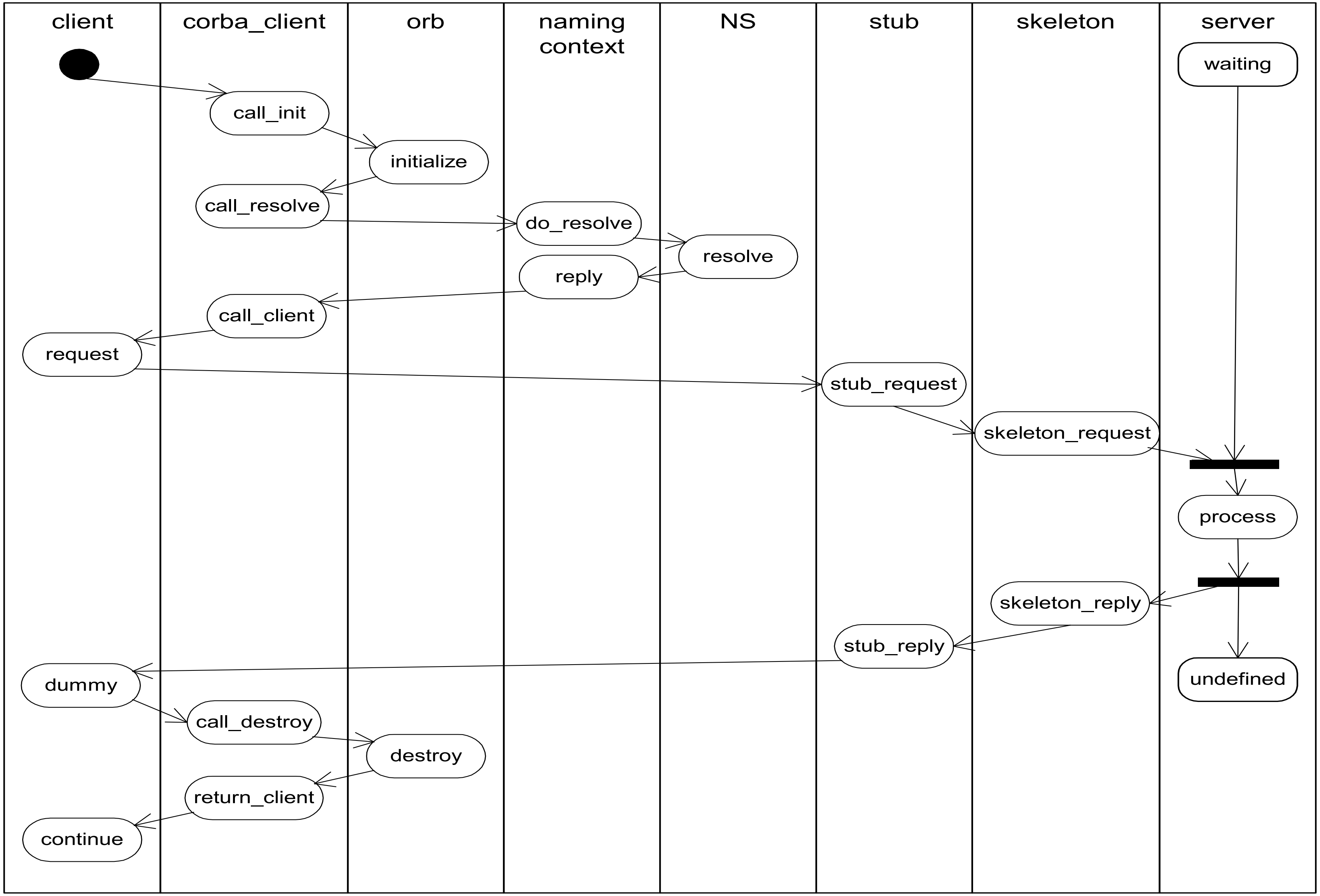
Transformation: activity, call



Transformation: activity, destroy



Transformation: activity complete



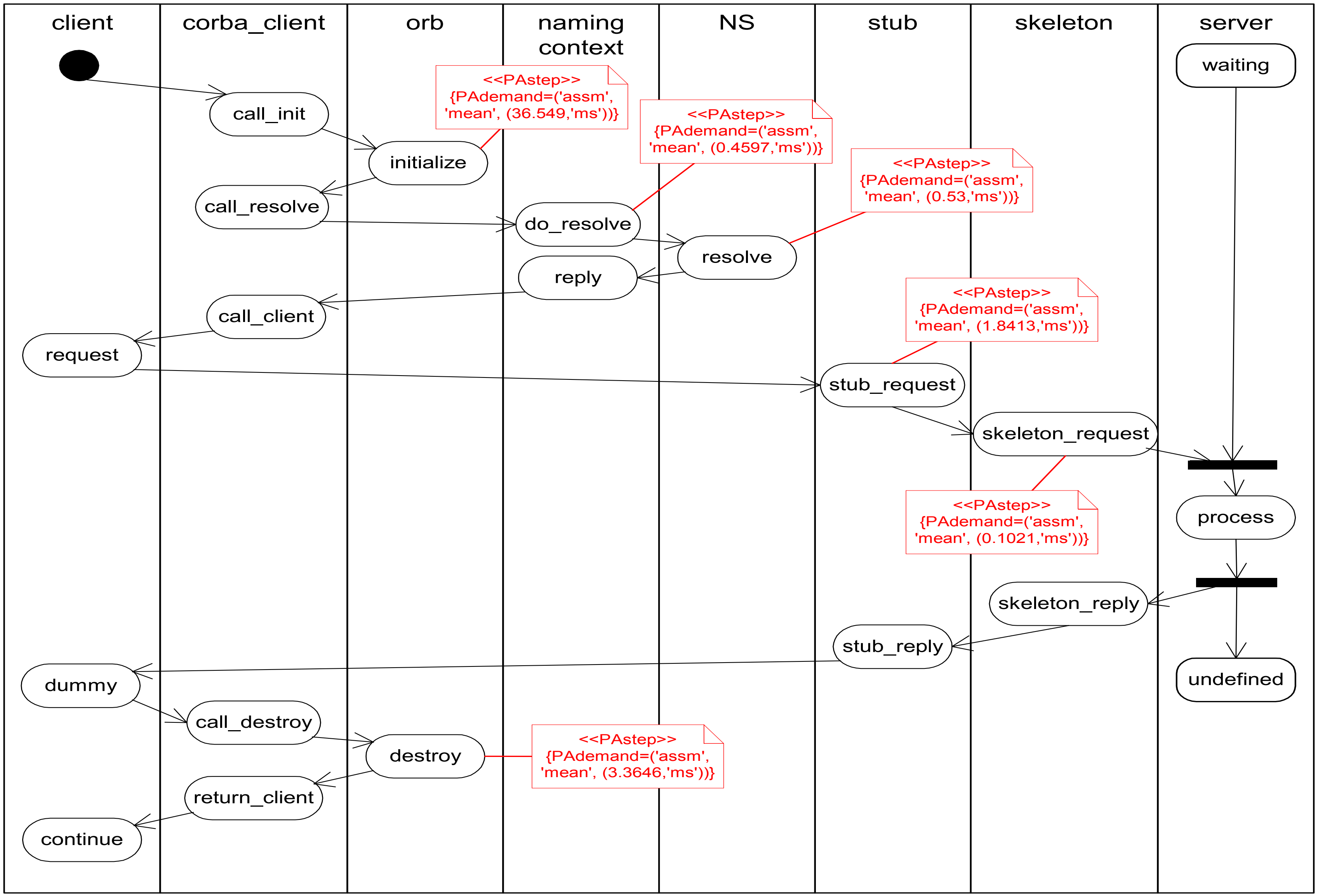
Transformation: performance

- Performance modelling using performance profile
- Actions get the `<<Pastep>>` stereotype
- Execution times: `Pademand` tagged value

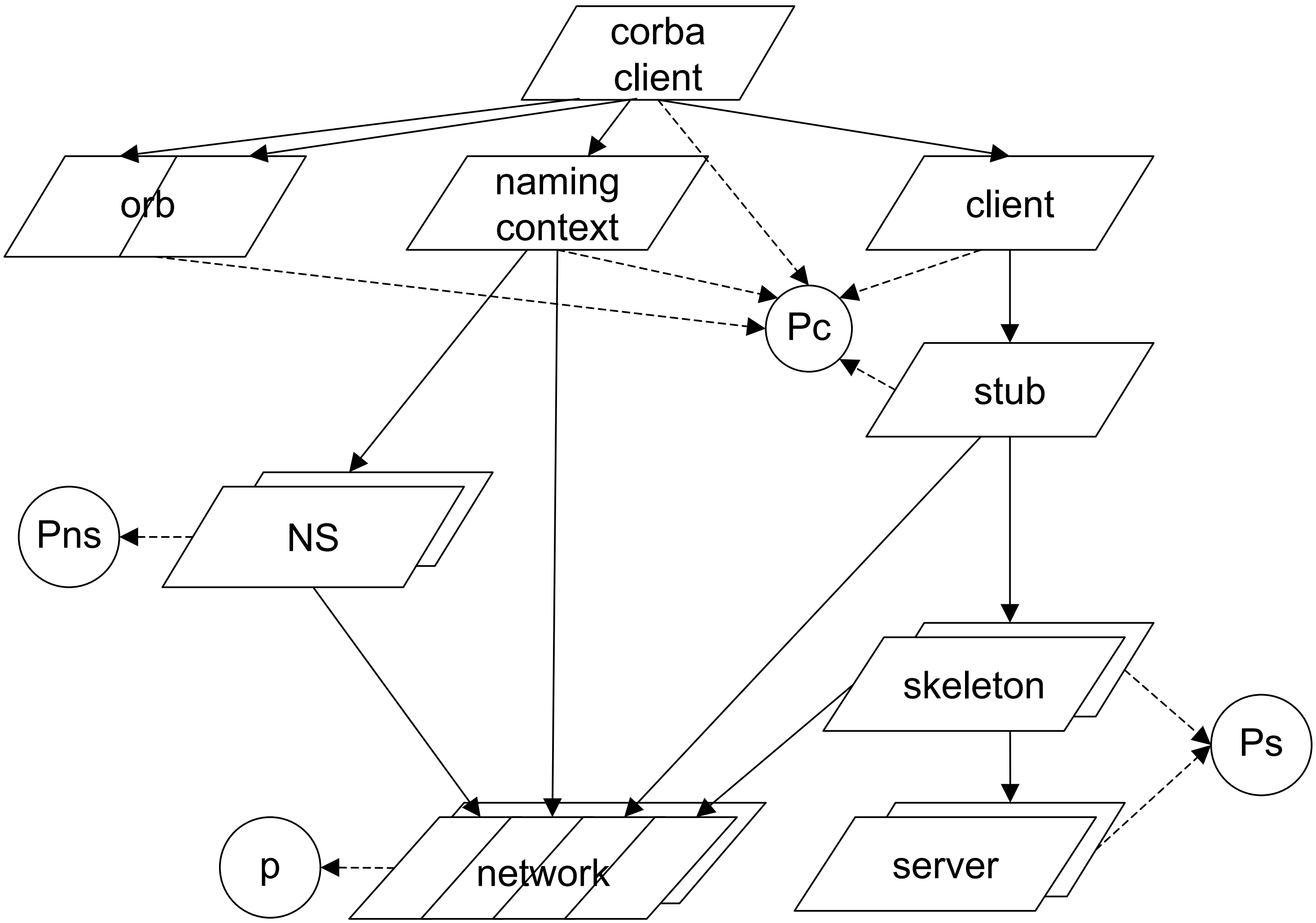
```
<middleware>
   <link id="link1" type="CORBA" NSref="NS1"  inittime="36.55"
    nscalltime="0.45" destroytime="3.36">
      <call cref="G.9" stubtime="1.84" skeletontime="0.10" />
   </link>

   <NS id="NS1" host="xmi.17" lookuptime="0.53" />
</middleware>
```

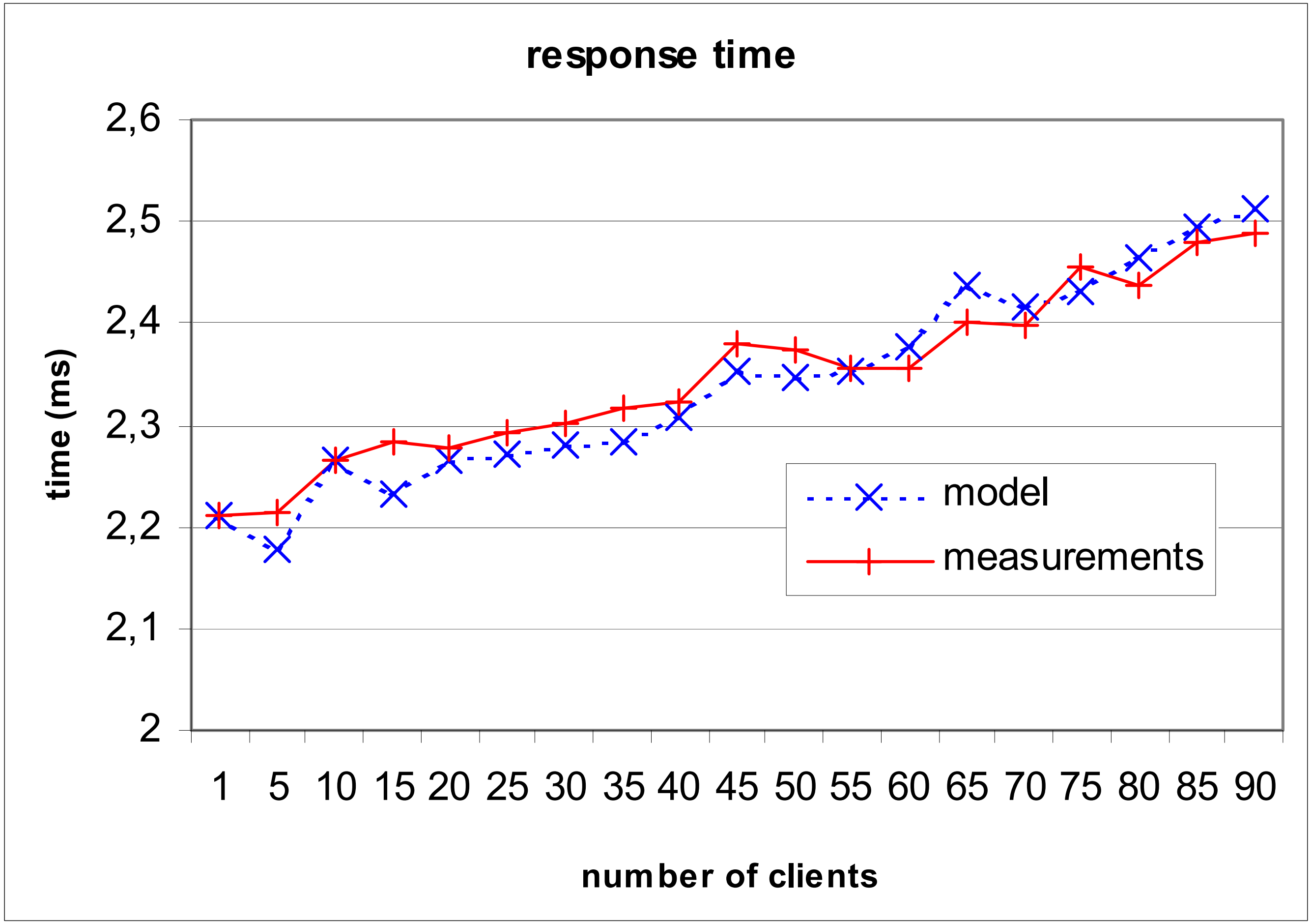
Transformation: activity final



The way forward



Test results: response time



Overview

- Software Performance Engineering
- SPE + MDA
- Middleware
- Transformation
- Conclusion

Conclusion

- Automatic inclusion of CORBA performance information during PIM-to-PSM transformation
- Allows early assessment of the performance impact of using CORBA, without knowledge of the CORBA internals
- Easily adaptable to other middleware types (allowing the designers to compare several types)