Modeling and detection of Performance Antipatterns in UML

Dipartimento di Informatica
Università degli studi di L'Aquila

PhD student
Catia Trubiani
catia.trubiani@univaq.it

Advisor
Vittorio Cortellessa
vittorio.cortellessa@univaq.it

Progetto PRIN PaCo (Performability-aware Computing)
L'Aquila, 02-03 marzo 2010

Roadmap

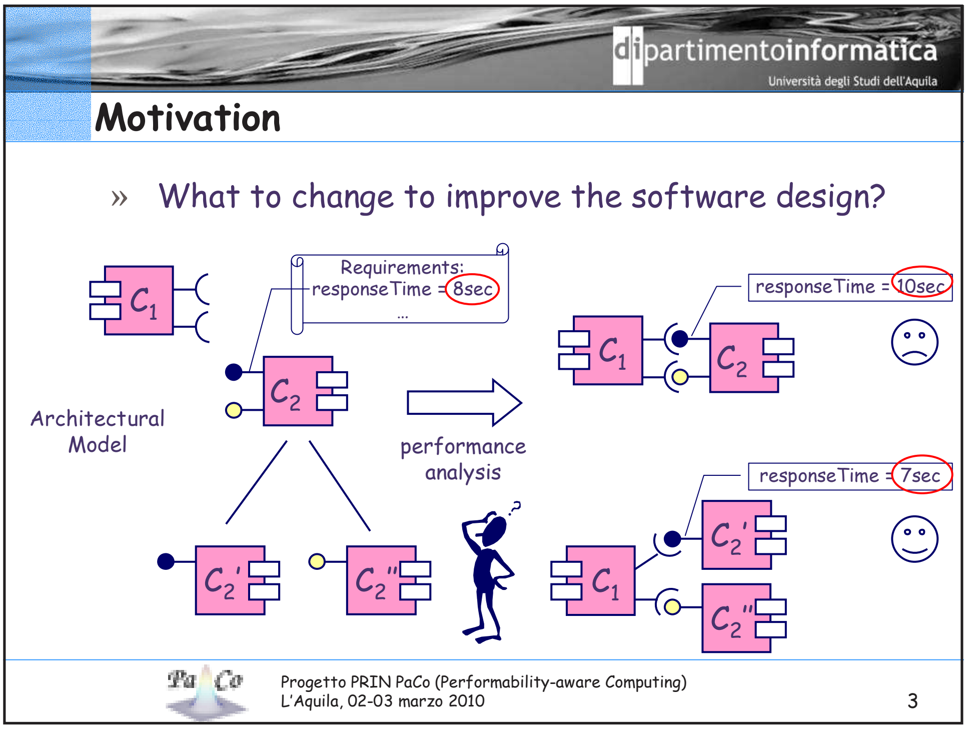
» Motivation

» Problem statement

» UML context

- Our approach
- A case study

» Ongoing and future works



Motivation

» What to change to improve the software design?

C_1

Requirements:
responseTime = 8sec
...

C_2

Architectural Model

performance analysis

C_2'

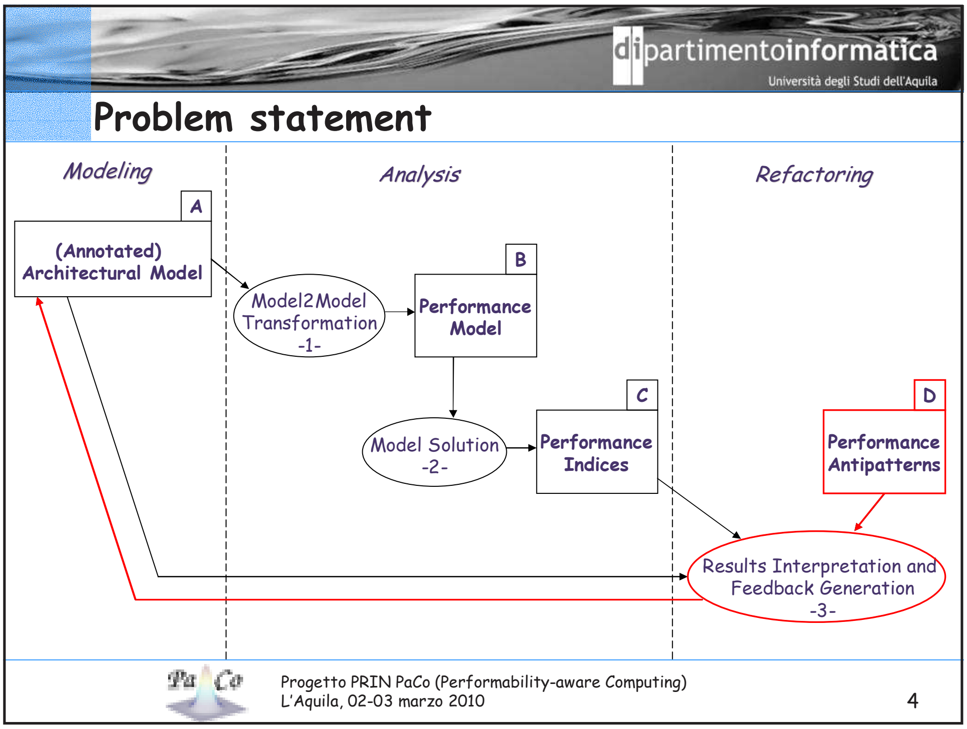
C_2''

responseTime = 10sec

C_1 C_2

responseTime = 7sec

C_1 C_2' C_2''



Problem statement

Modeling

Analysis

Refactoring

A

(Annotated) Architectural Model

Model2Model Transformation -1-

B

Performance Model

Model Solution -2-

C

Performance Indices

D

Performance Antipatterns

Results Interpretation and Feedback Generation -3-

(Performance) Antipatterns

» W.J.Brown, R.C. Malveau, H.W. Mc Cornich III, and T.J. Mowbray. "Antipatterns: Refactoring Software, Architectures, and Project in Crisis", 1998.

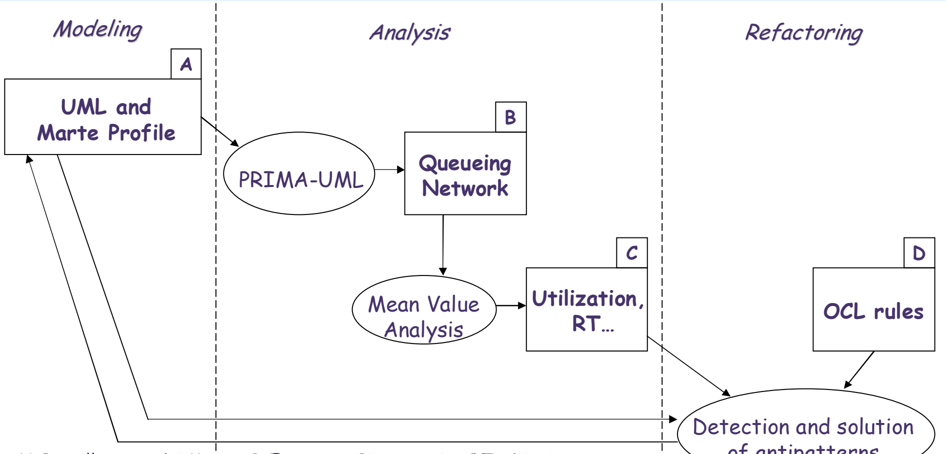
-Look at negative features of a software system:

>The definition includes common mistakes (i.e. "Bad practice") in software development as well as their solutions

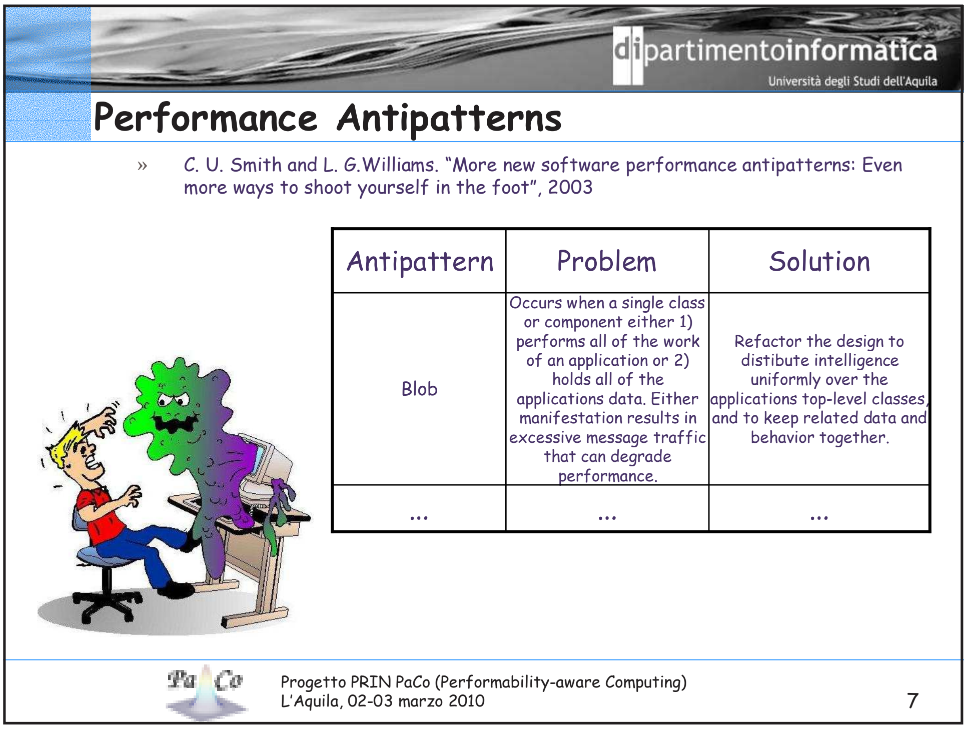
>Conceptually similar to "Design Patterns": recurring solutions to common design problems

» What to avoid and how to solve (performance) problems!

Our approach in the UML context



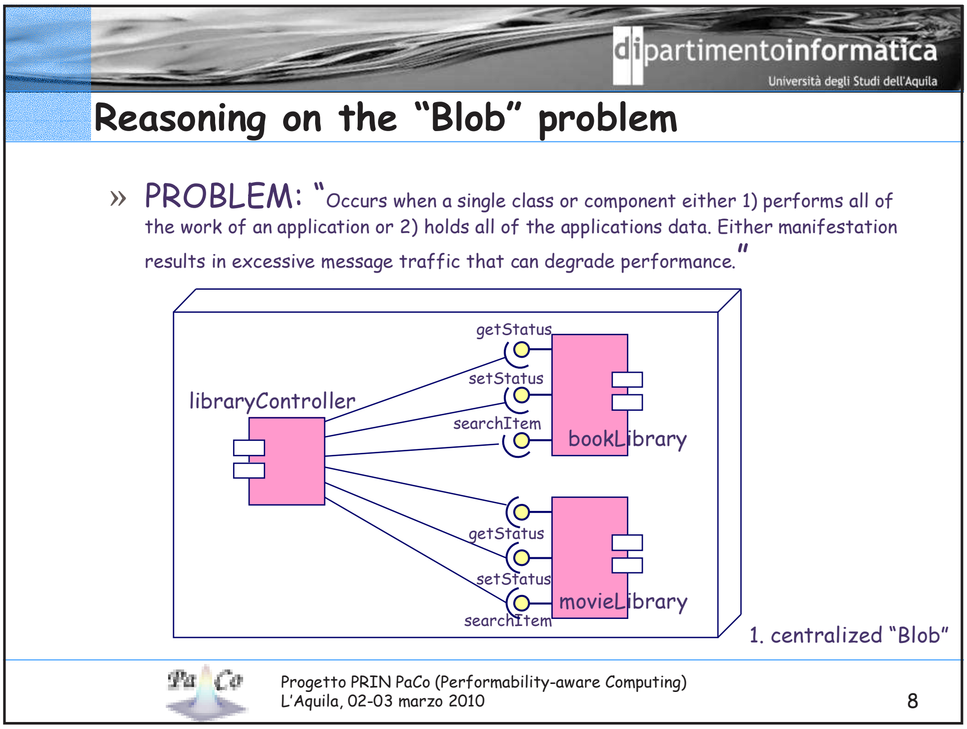
V.Cortellessa, A.Di Marco, R.Eramo, A. Pierantonio, C.Trubiani "Digging into UML models to remove Performance Antipatterns", to appear in the proceedings of ICSE Companion 2010.



Performance Antipatterns

» C. U. Smith and L. G.Williams. "More new software performance antipatterns: Even more ways to shoot yourself in the foot", 2003

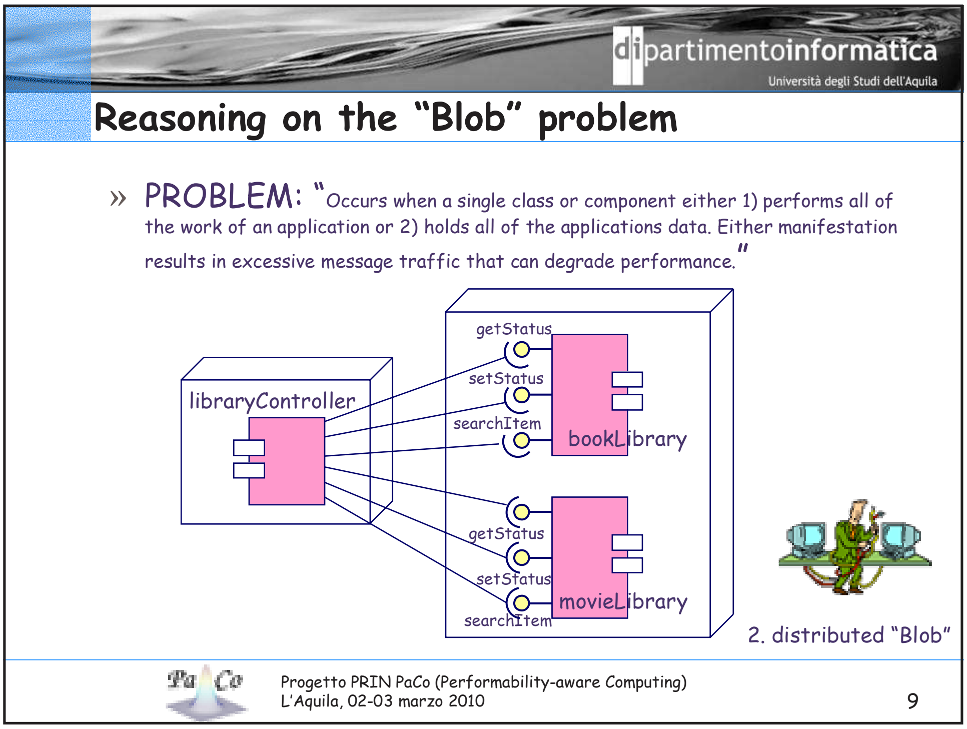
Antipattern	Problem	Solution
Blob	Occurs when a single class or component either 1) performs all of the work of an application or 2) holds all of the applications data. Either manifestation results in excessive message traffic that can degrade performance.	Refactor the design to distibute intelligence uniformly over the applications top-level classes, and to keep related data and behavior together.
...	...	...



Reasoning on the "Blob" problem

» PROBLEM: "Occurs when a single class or component either 1) performs all of the work of an application or 2) holds all of the applications data. Either manifestation results in excessive message traffic that can degrade performance."

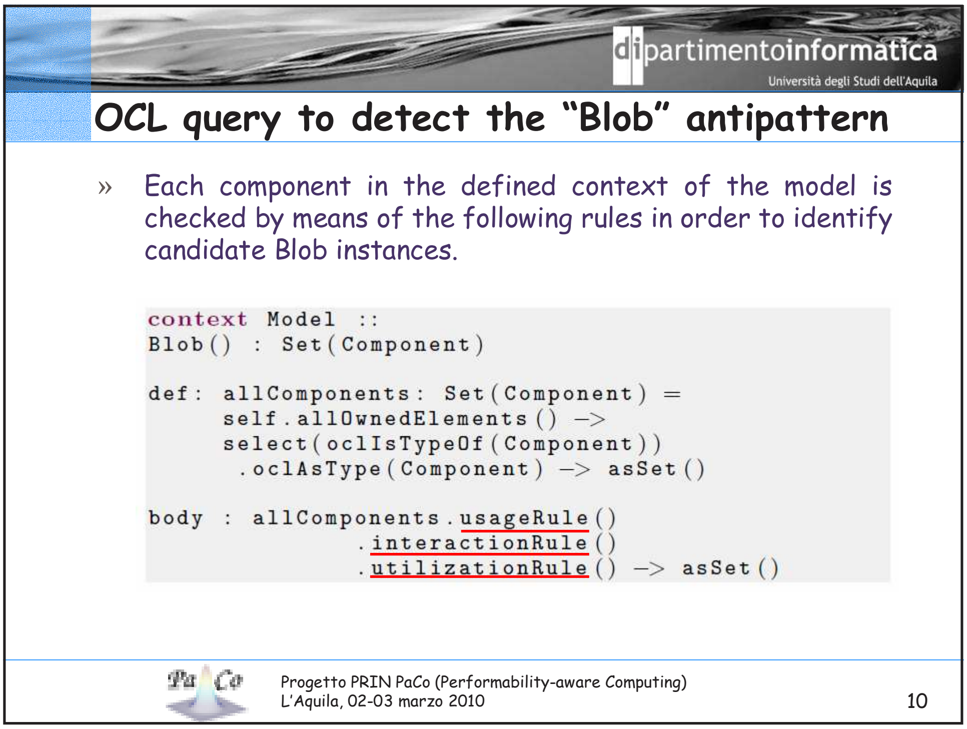
1. centralized "Blob"



Reasoning on the "Blob" problem

» PROBLEM: "Occurs when a single class or component either 1) performs all of the work of an application or 2) holds all of the applications data. Either manifestation results in excessive message traffic that can degrade performance."

2. distributed "Blob"



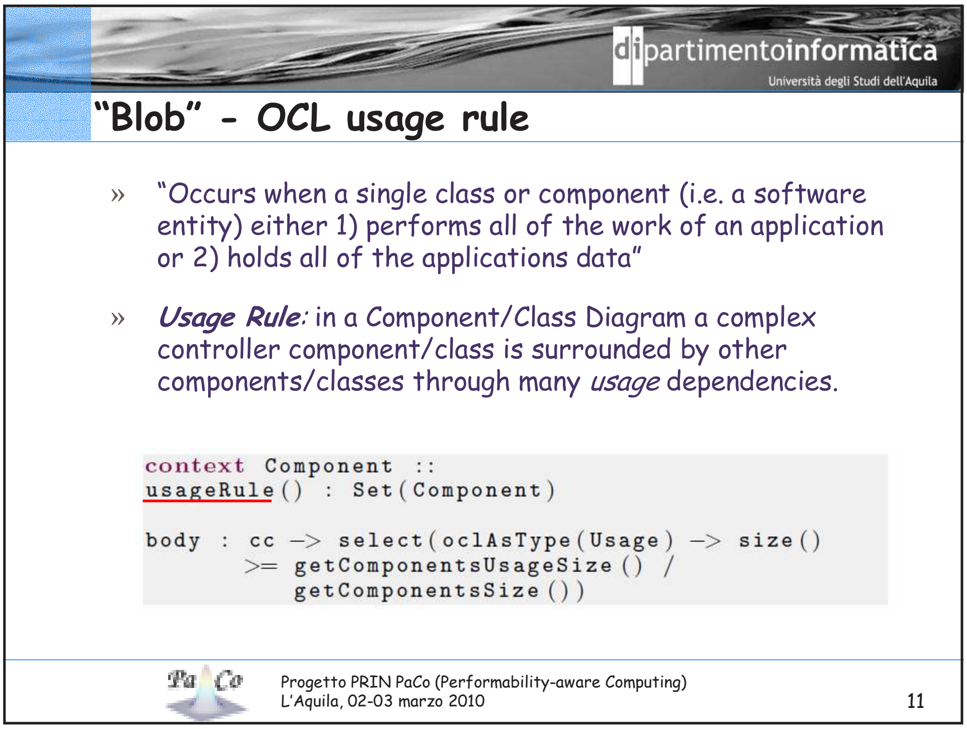
OCL query to detect the "Blob" antipattern

» Each component in the defined context of the model is checked by means of the following rules in order to identify candidate Blob instances.

```
context Model ::
Blob() : Set(Component)

def: allComponents: Set(Component) =
     self.allOwnedElements() ->
     select(oclIsTypeOf(Component))
      .oclAsType(Component) -> asSet()

body : allComponents.usageRule()
              .interactionRule()
              .utilizationRule() -> asSet()
```

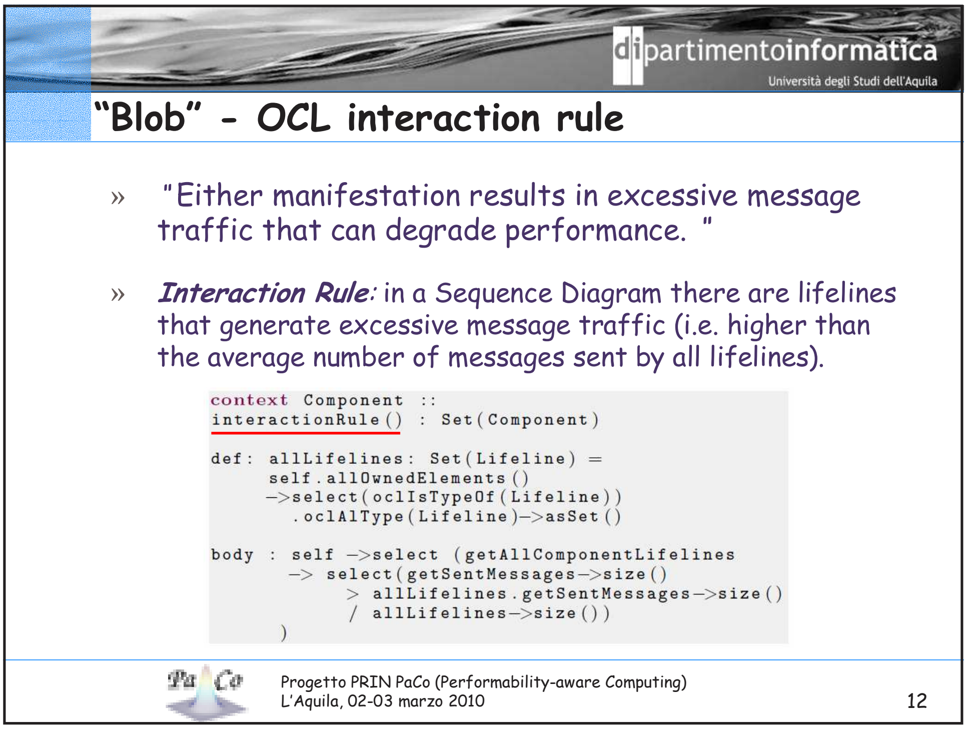


"Blob" - OCL usage rule

- "Occurs when a single class or component (i.e. a software entity) either 1) performs all of the work of an application or 2) holds all of the applications data"
- *Usage Rule:* in a Component/Class Diagram a complex controller component/class is surrounded by other components/classes through many *usage* dependencies.

```
context Component ::
usageRule() : Set(Component)

body : cc -> select(oclAsType(Usage) -> size()
        >= getComponentsUsageSize() /
           getComponentsSize())
```



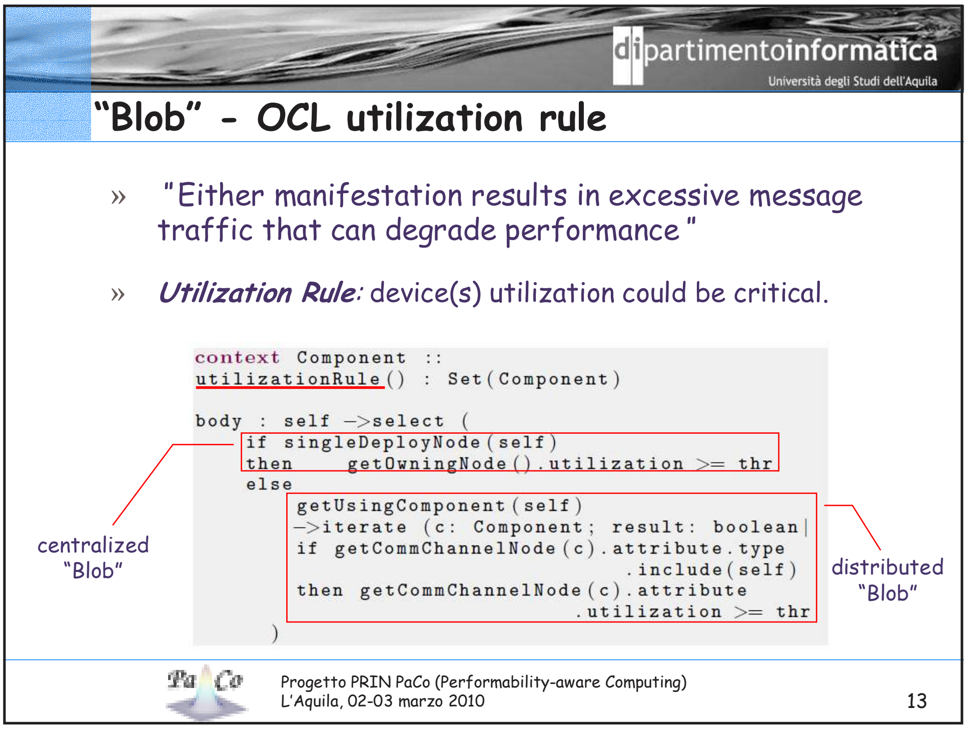
"Blob" - OCL interaction rule

- "Either manifestation results in excessive message traffic that can degrade performance. "
- ***Interaction Rule:*** in a Sequence Diagram there are lifelines that generate excessive message traffic (i.e. higher than the average number of messages sent by all lifelines).

```
context Component ::
interactionRule() : Set(Component)

def: allLifelines: Set(Lifeline) =
     self.allOwnedElements()
     ->select(oclIsTypeOf(Lifeline))
       .oclAlType(Lifeline)->asSet()

body : self ->select (getAllComponentLifelines
       -> select(getSentMessages->size()
            > allLifelines.getSentMessages->size()
            / allLifelines->size())
      )
```



"Blob" - OCL utilization rule

» "Either manifestation results in excessive message traffic that can degrade performance"

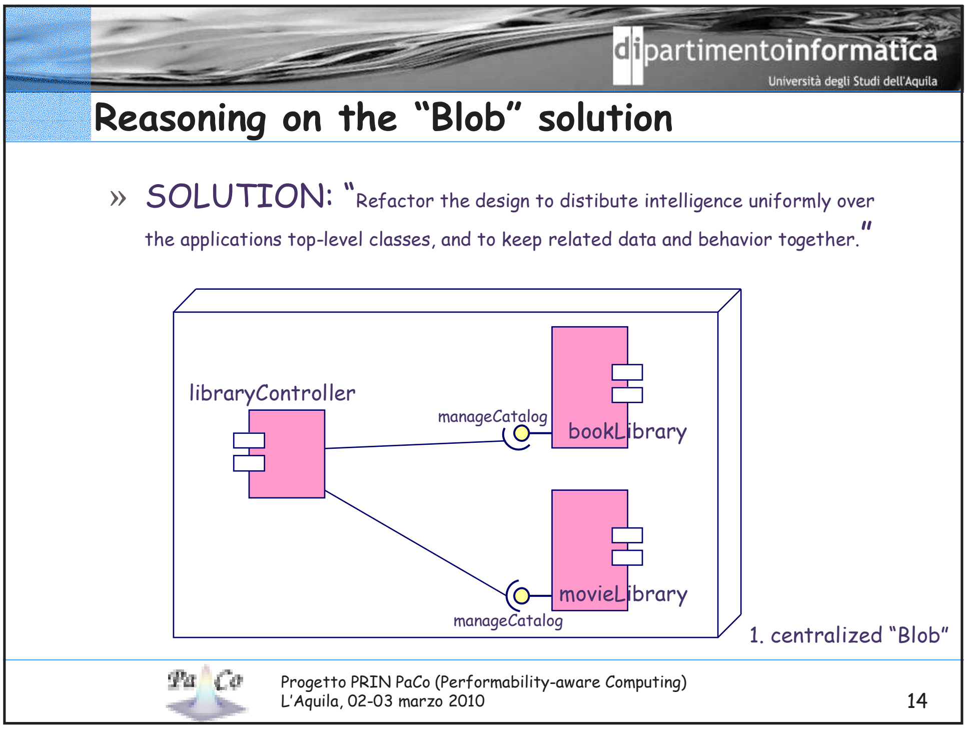
» *Utilization Rule:* device(s) utilization could be critical.

```
context Component ::
utilizationRule() : Set(Component)

body : self ->select (
    if singleDeployNode(self)
    then    getOwningNode().utilization >= thr
    else
        getUsingComponent(self)
        ->iterate (c: Component; result: boolean|
        if getCommChannelNode(c).attribute.type
                              .include(self)
        then getCommChannelNode(c).attribute
                            .utilization >= thr
    )
```

centralized "Blob"

distributed "Blob"



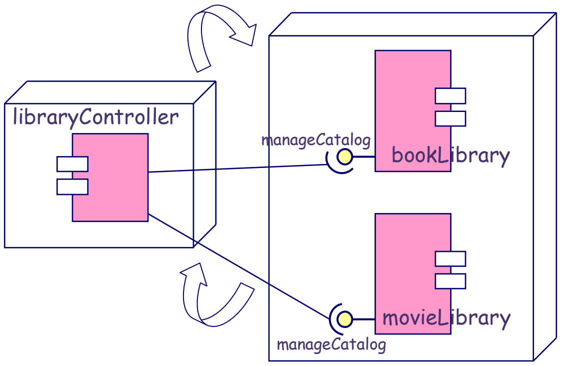
Reasoning on the "Blob" solution

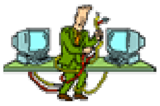
» SOLUTION: "Refactor the design to distibute intelligence uniformly over the applications top-level classes, and to keep related data and behavior together."

1. centralized "Blob"

Reasoning on the "Blob" solution

» SOLUTION: "Refactor the design to distibute intelligence uniformly over the applications top-level classes, and to keep related data and behavior together."

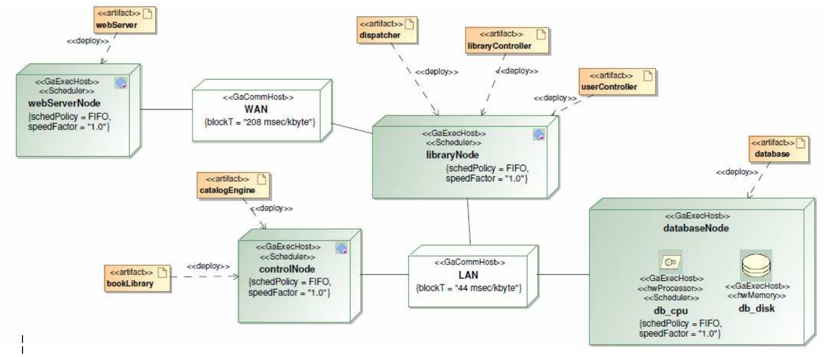


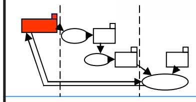


2. distributed "Blob"

A case study in the UML context

» Modeling the Electronic Commerce System (ECS)





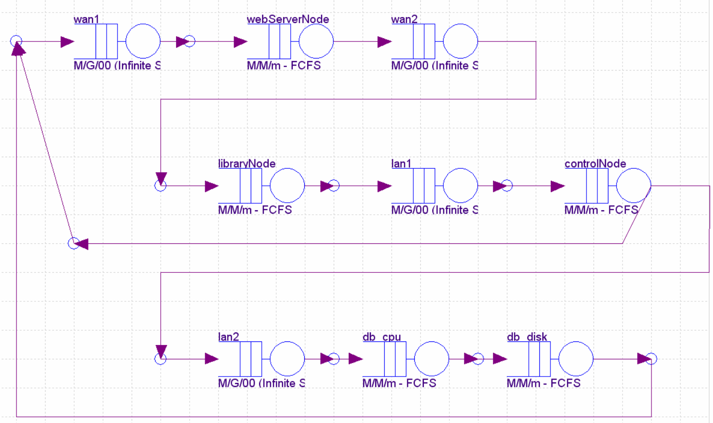
UML Deployment Diagram

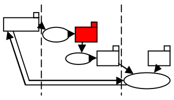
ECS - Performance Analysis

» Performance Model: Queueing Networks

wan	1040 msec
lan	**396** msec
$webServerNode$	4 msec
$libraryNode$	**9** msec
$controlNode$	**6** msec
$databaseNode_{cpu}$	15 msec
$databaseNode_{disk}$	30 msec

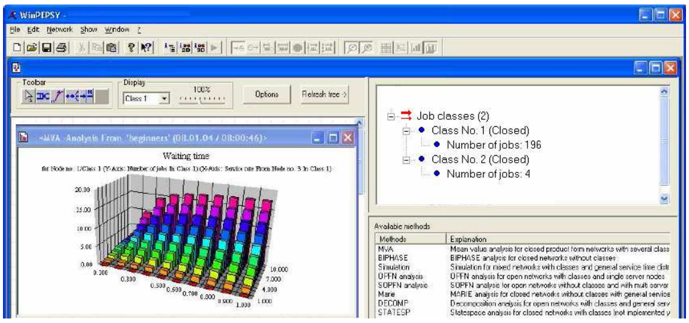
input parameters





ECS - Performance Analysis

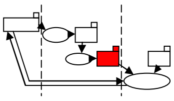
» Performance Indices: e.g. Response Time (RT).


WinPEPSY -
File Edit Network Show Window ?
Toolbar
Display
Class 1
100%
Options
Refresh tree ->
<MVA -Analysis From 'beginners' (08.01.04 / 08:00:46)>
Waiting time
for Node no. 1/Class 1 (Y-Axis: Number of jobs in Class 1) (X-Axis: Service rate From Node no. 3 in Class 1)
20.00
15.00
10.00
5.00
0.00
0.100
0.300
0.500
0.600
0.700
0.800
0.900
1.000
10.000
7.000
4.000
1.000

Job classes (2)
- Class No. 1 (Closed)
  - Number of jobs: 196
- Class No. 2 (Closed)
  - Number of jobs: 4

Available methods

Methods	Explanation
MVA	Mean value analysis for closed product form networks with several class
BIPHASE	BIPHASE analysis for closed networks without classes
Simulation	Simulation for mixed networks with classes and general service time distr
OPFN analysis	OPFN analysis for open networks with classes and single server nodes
SOPFN analysis	SOPFN analysis for open networks without classes and with multi server
Marie	MARIE analysis for closed networks without classes with general service
DECOMP	Decomposition analysis for open networks with classes and general serv
STATESP	Statespace analysis for closed networks with classes (not implemented y



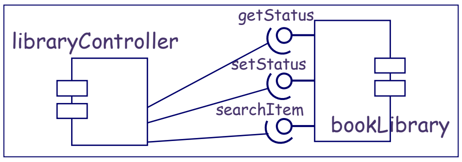
Requirement	Required Value	Observed Value
RT(browseCatalog)	1.5 sec	1.61 sec

ECS – Detecting antipatterns

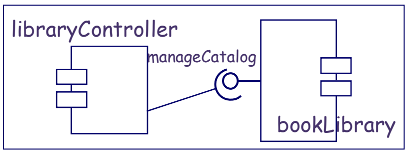
Antipattern	Problem	Solution
Blob	libraryController performs most of the work, it generates excessive message traffic.	Refactor the design to keep related data and behavior together. Delegate some work from libraryController to bookLibrary.
Empty Semi Trucks	An excessive number of requests is required to perform the task of registering new users.	Refactor the design combining items into messages to make better use of available bandwidth
Concurrent Processing Systems	Processing cannot make use of the processor webServerNode.	Restructure software or changing scheduling algorithms between processors libraryNode and webServerNode.

ECS – Refactoring (1/2)

» Solving the "Blob" antipattern



ECS

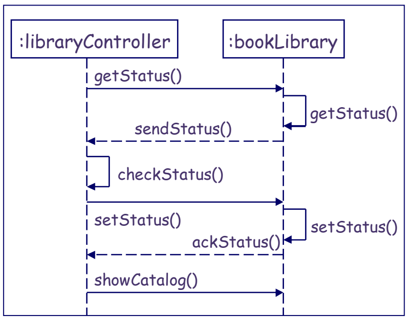


ECS\{Blob}

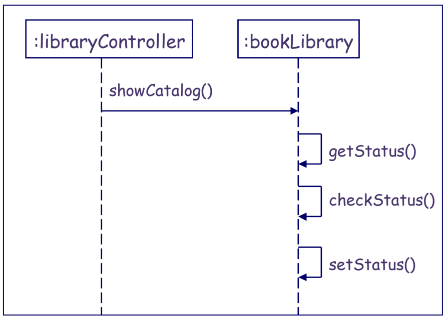
UML Component Diagram

ECS – Refactoring (2/2)

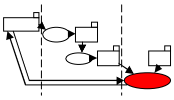
» Solving the "Blob" antipattern



ECS



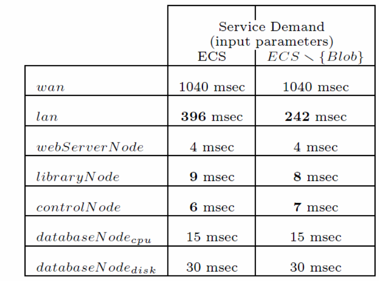
ECS\{Blob}



UML Sequence Diagram

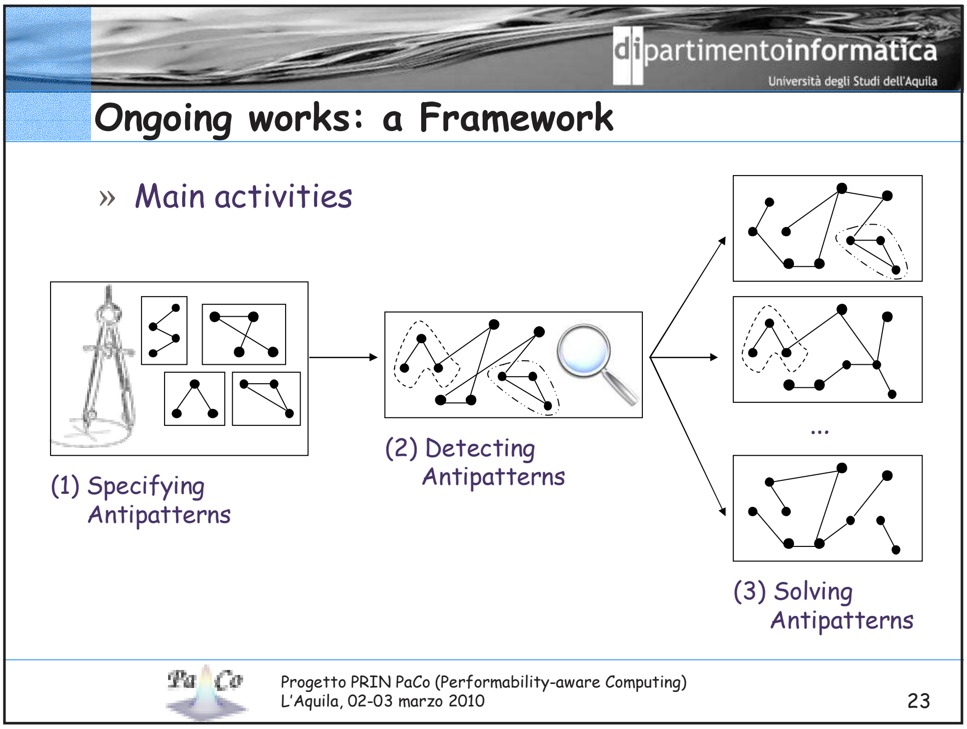
ECS\{Blob} system: performance results

» Performance Analysis of the ECS\{Blob} system



	Service Demand (input parameters)	
	ECS	$ECS \setminus \{Blob\}$
wan	1040 msec	1040 msec
lan	**396** msec	**242** msec
$webServerNode$	4 msec	4 msec
$libraryNode$	**9** msec	**8** msec
$controlNode$	**6** msec	**7** msec
$databaseNode_{cpu}$	15 msec	15 msec
$databaseNode_{disk}$	30 msec	30 msec

Requirement	Required Value	ECS Observed Value	ECS\{blob} Observed Value
RT(browseCatalog)	1.5 sec	1.61 sec	1.44 sec



Ongoing works: a Framework

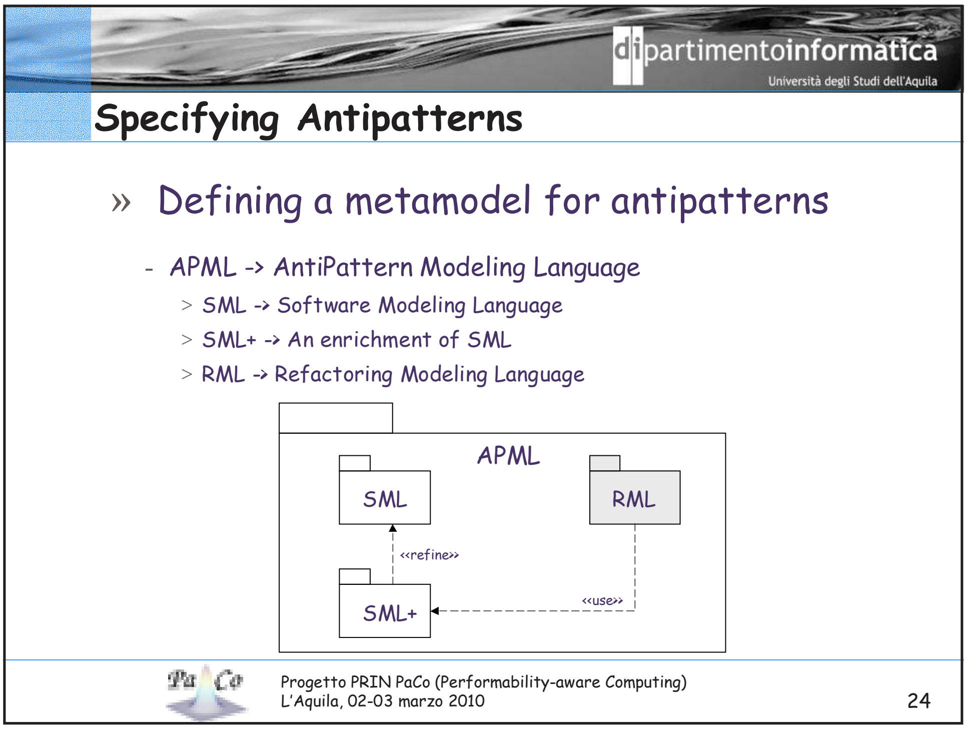
» Main activities

(1) Specifying Antipatterns

(2) Detecting Antipatterns

...

(3) Solving Antipatterns



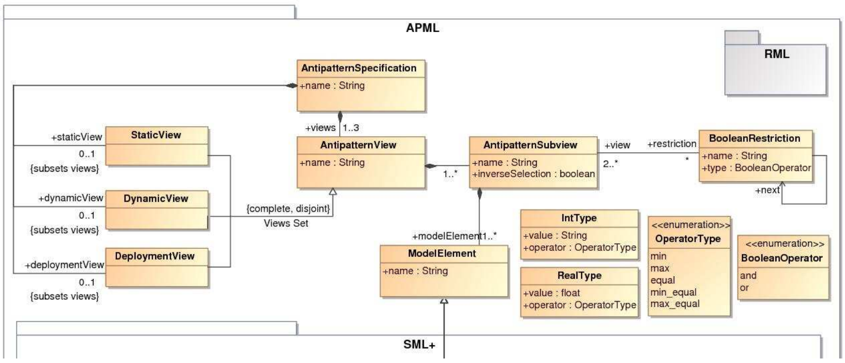
Specifying Antipatterns

» Defining a metamodel for antipatterns

- APML -> AntiPattern Modeling Language
  - > SML -> Software Modeling Language
  - > SML+ -> An enrichment of SML
  - > RML -> Refactoring Modeling Language

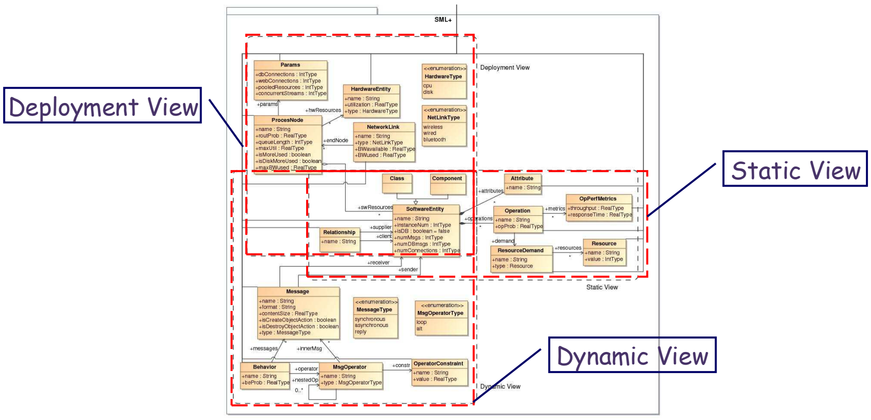
Metamodel for antipatterns

» An excerpt of APML



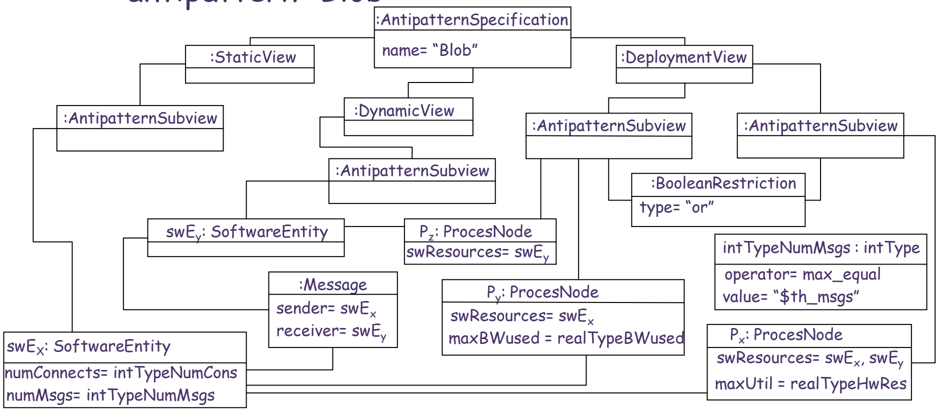
Metamodel for antipatterns

» Software Modeling Language SML+



Modeling with APML

» An example: how to model the performance antipattern "Blob"



:AntipatternSpecification
name= "Blob"

:StaticView

:DeploymentView

:DynamicView

:AntipatternSubview

:AntipatternSubview

:AntipatternSubview

:AntipatternSubview

:BooleanRestriction
type= "or"

swE$_y$: SoftwareEntity

P$_z$: ProcesNode
swResources= swE$_y$

intTypeNumMsgs : intType
operator= max_equal
value= "$th_msgs"

:Message
sender= swE$_x$
receiver= swE$_y$

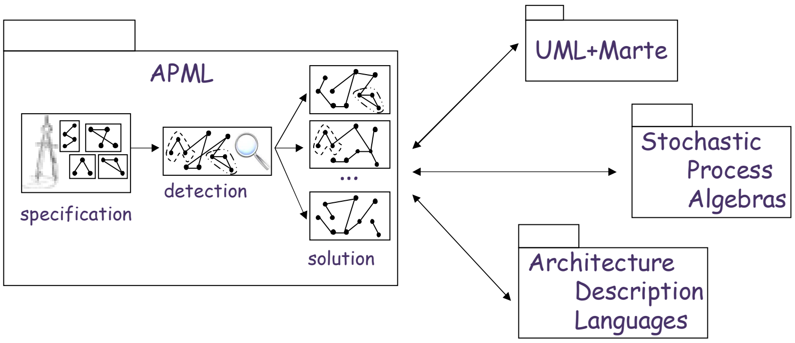
P$_y$: ProcesNode
swResources= swE$_x$
maxBWused = realTypeBWused

P$_x$: ProcesNode
swResources= swE$_x$, swE$_y$
maxUtil = realTypeHwRes

swE$_X$: SoftwareEntity
numConnects= intTypeNumCons
numMsgs= intTypeNumMsgs

Goal: Antipatterns across different languages

» Validate the scope of the whole approach to assess the indipendence of any concrete notation.



APML

specification

detection

...

solution

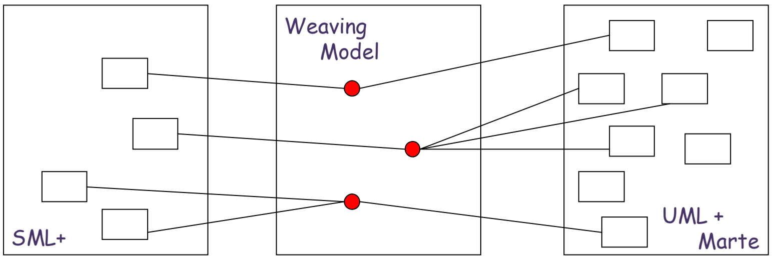
UML+Marte

Stochastic Process Algebras

Architecture Description Languages

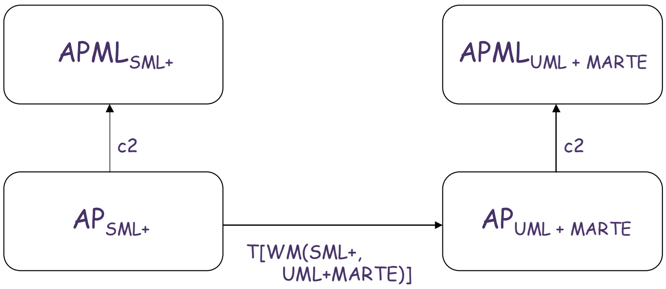
Our metamodel in concrete notations

» Weaving Models to drive correspondences between our SML+ and a concrete modeling notation (e.g. UML + Marte).



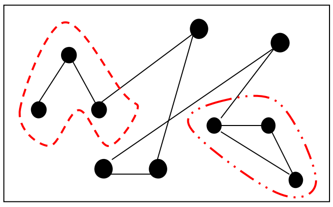
Antipatterns in concrete notations

» High-order Transformations to drive correspondences between the antipattern model in our SML+ and a concrete modeling notation (e.g. UML + Marte).



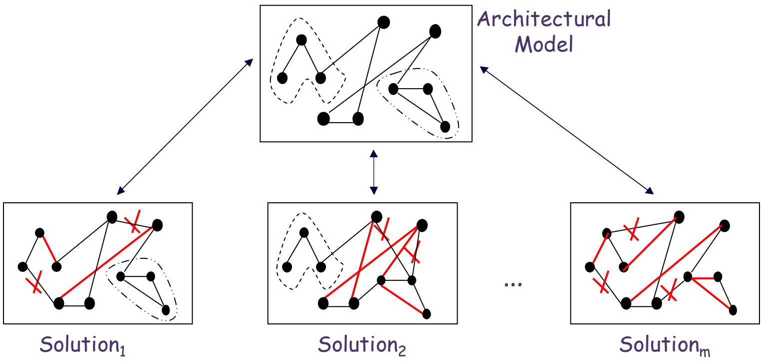
Detecting antipatterns

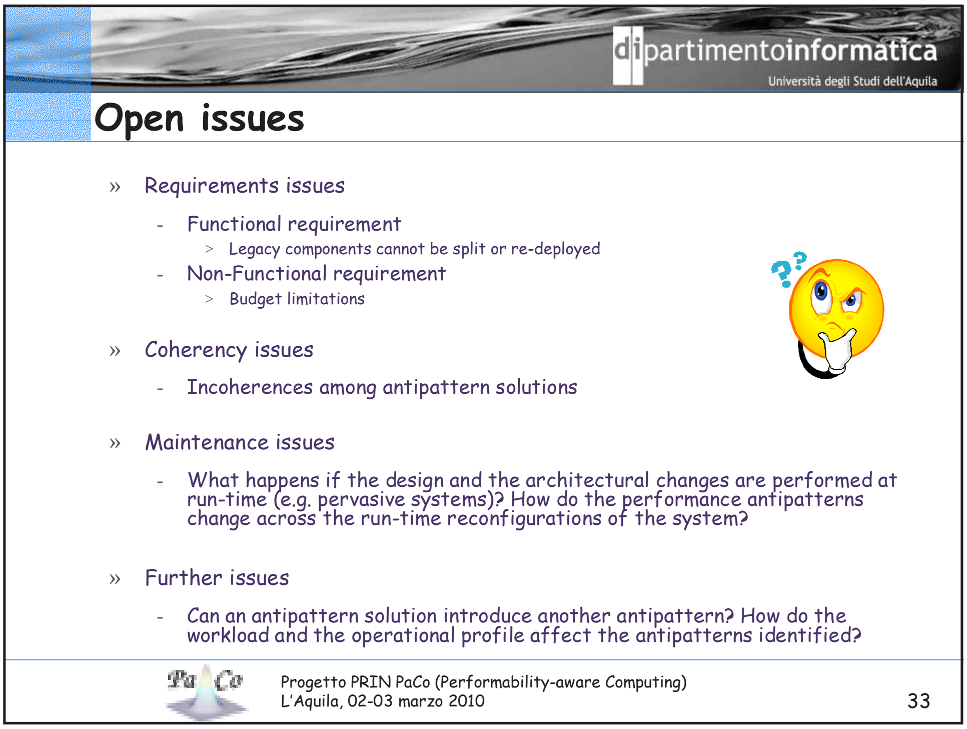
» Translating the occurrence of antipatterns with OCL expressions.



Future works: solving Antipatterns

» Apply concepts of difference models.





Open issues

- » Requirements issues
  - Functional requirement
    - > Legacy components cannot be split or re-deployed
  - Non-Functional requirement
    - > Budget limitations
- » Coherency issues
  - Incoherences among antipattern solutions
- » Maintenance issues
  - What happens if the design and the architectural changes are performed at run-time (e.g. pervasive systems)? How do the performance antipatterns change across the run-time reconfigurations of the system?
- » Further issues
  - Can an antipattern solution introduce another antipattern? How do the workload and the operational profile affect the antipatterns identified?