dipartimentoinformatica
Università degli Studi dell'Aquila

SFM-12:MDE 12th International School on Formal Methods for the Design of Computer, Communication and Software Systems: Model-Driven Engineering
Bertinoro, Italy, 18-23 June 2012

Software Performance Antipatterns: Modeling and Analysis

Vittorio Cortellessa, Antinisca Di Marco, Catia Trubiani
Computer Science Department, University of L'Aquila, Italy

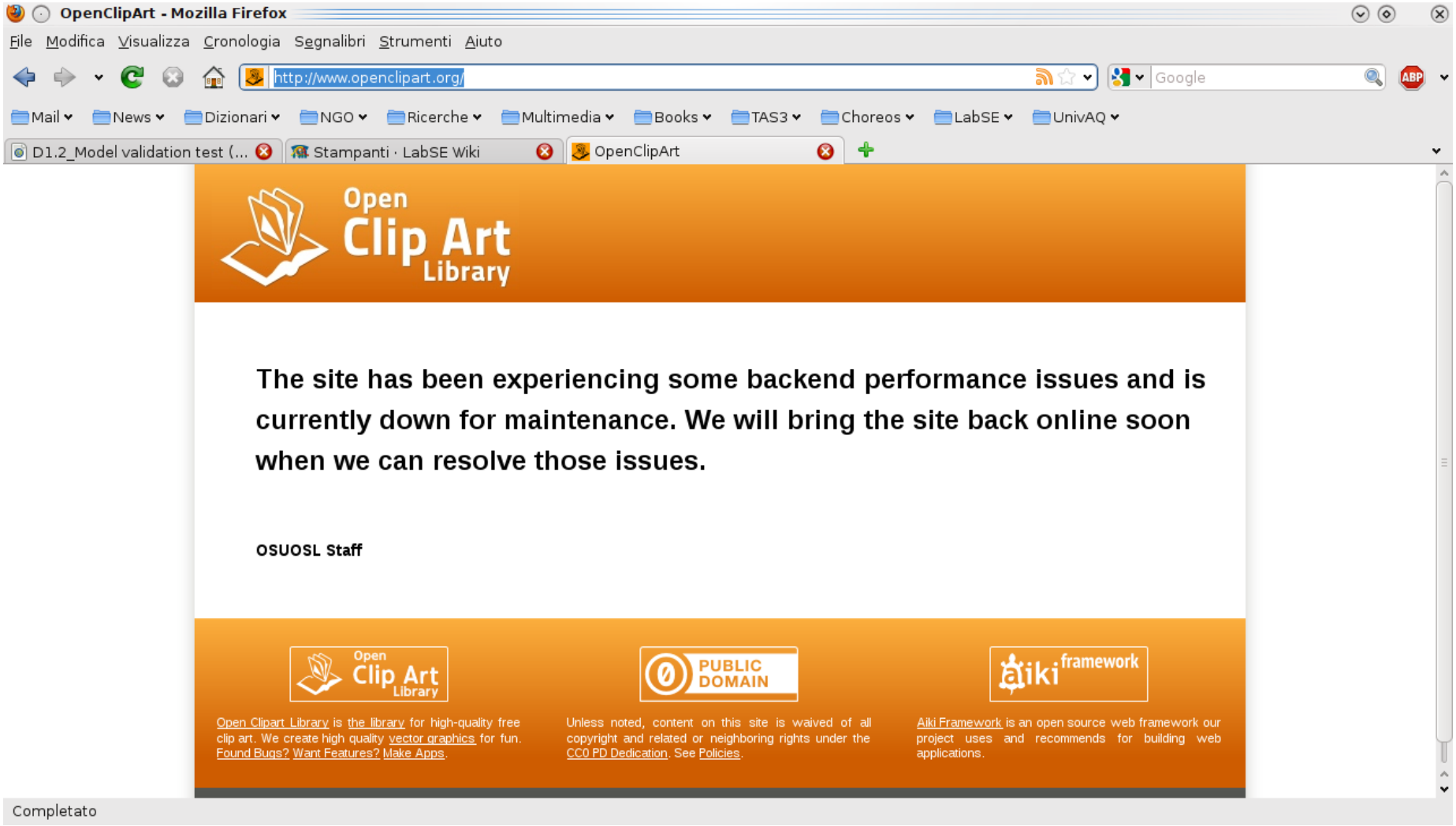
Credits for this course go to…

Catia Trubiani
Antinisca Di Marco

Romina Eramo
Alfonso Pierantonio
Davide Arcelli

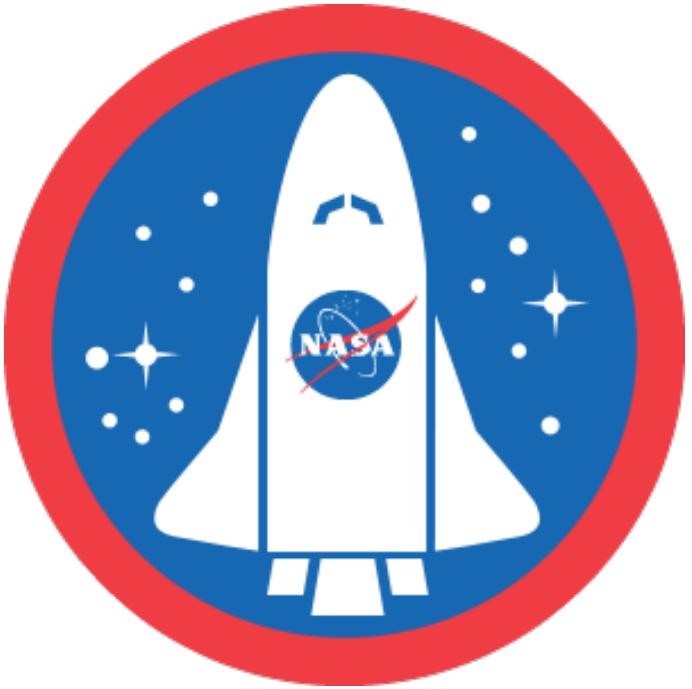
All my undergraduate and graduate students who have lost some of their health on antipatterns

Performance problems



Performance problems

- NASA was forced to **delay the launch** of a satellite for more than **eight months**



- The delay was caused because the Flight Operations Segment **software** had **unacceptable response times** for developing satellite schedules, and **poor performance** in analyzing satellite status and telemetry data

Performance problems – how tackled?!

Modeling

1970-1990

System performance modeling (QN, PN, …)

1990-2000

Software performance since early lifecycle phases

2000-2008

Model transformations for performance model generation

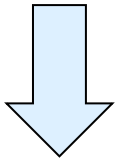
2008-today

Performance model evolution to support **adaptive software** to changes: requirements, context, …

Analysis

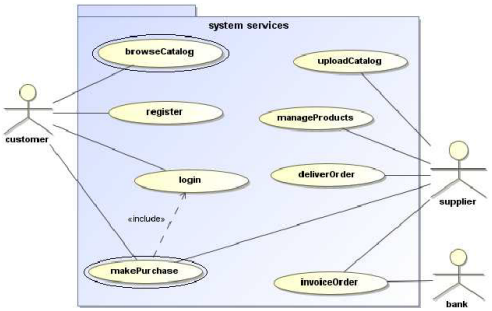
1970-today

Solve performance models with (ever more sophisticated!) tools

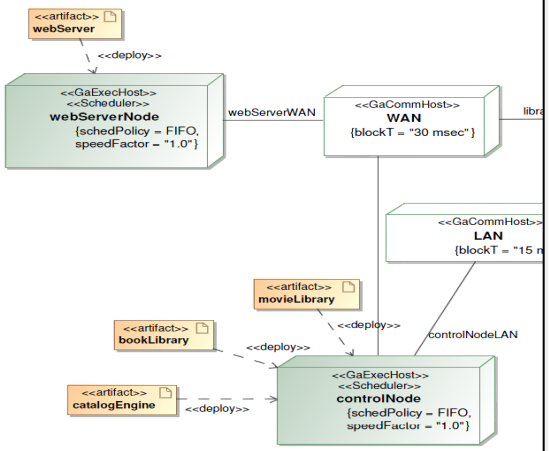


Bottleneck analysis

Software models



UML Use Case Diagram

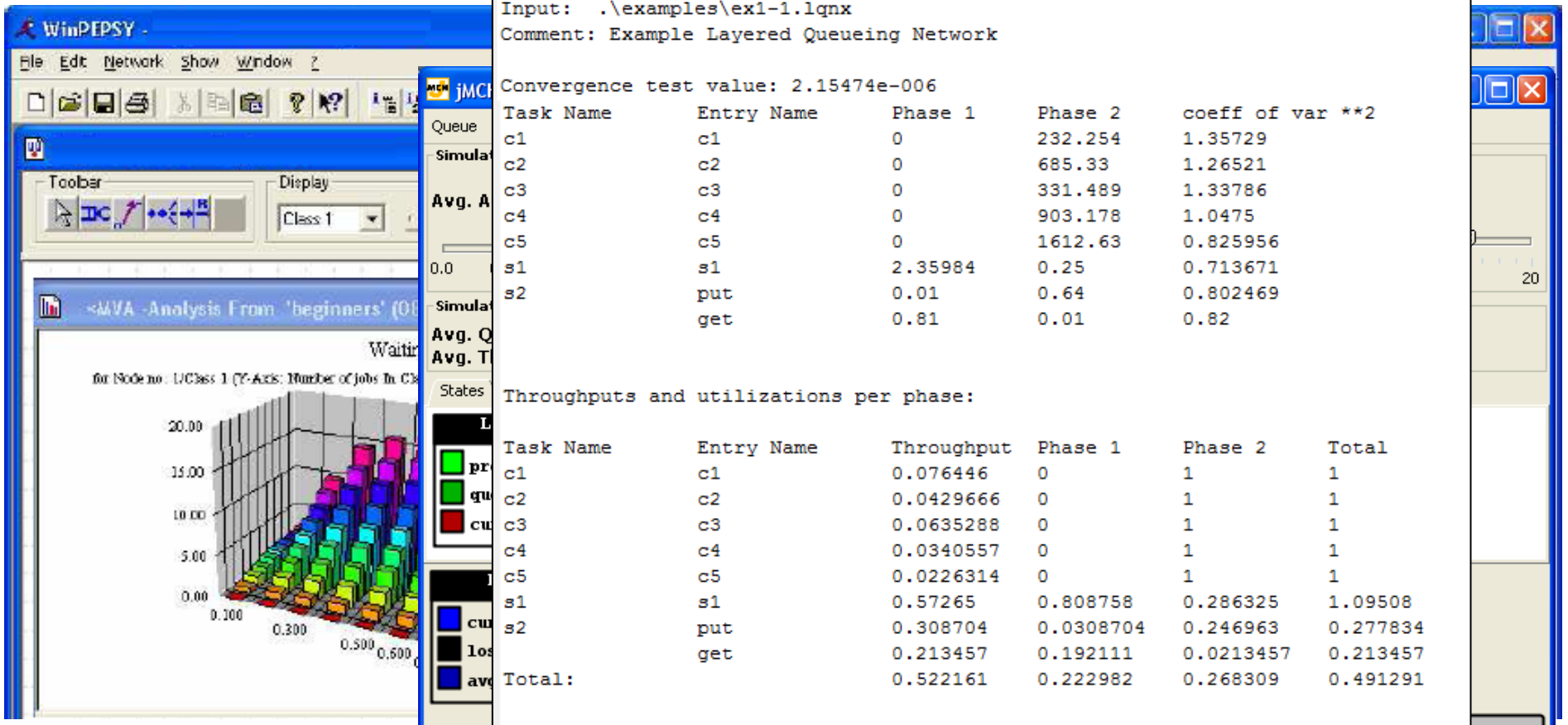


UML Deployment Diagram

```
ARCHI_TYPE boa( const integer ma_num := 5,
    const rate download_rate :=2441.40625,
    const rate upload_rate :=305.17578125,
    const rate balancer_rate_a :=20000000,
    const rate balancer_rate_b :=10000000,
    const rate server_req_rate:= 70000000,
    const rate server_result_rate:= 85995,
    const rate data_fetch_rate:= 36.585,
    const integer buffer_size :=10)


ARCHI_ELEM_TYPES

  ELEM_TYPE MA_Type(void)

      BEHAVIOR

      MobileApp(void; void) =
        <generate_best_path_req, inf> . <trasmit_req_best_path, inf>

      INPUT_INTERACTIONS

      UNI receive_best_path;
      generate_best_path_req

      OUTPUT_INTERACTIONS

      UNI trasmit_req_best_path
```

Architectural specification

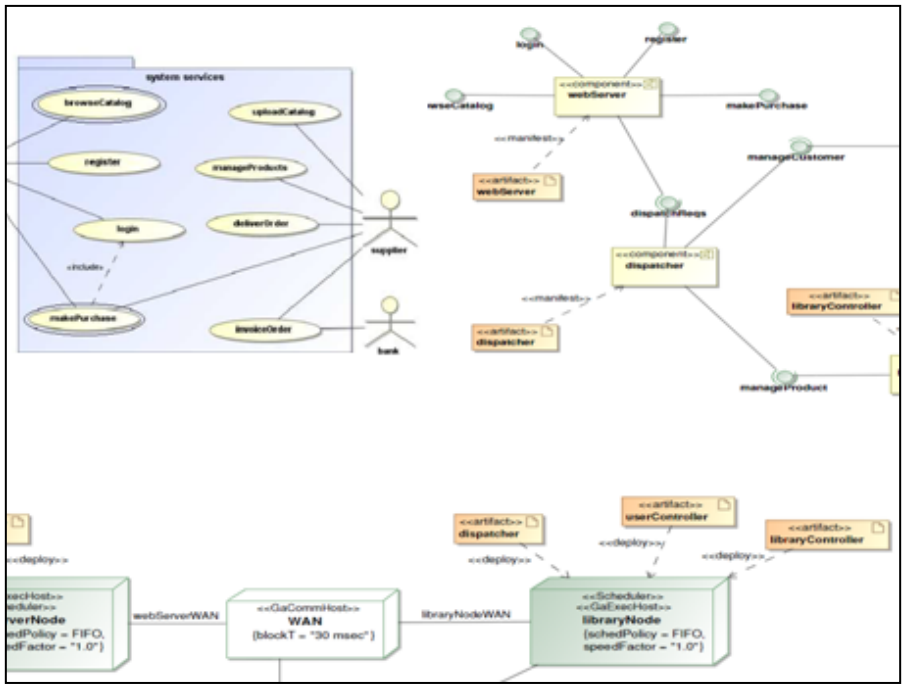
Performance

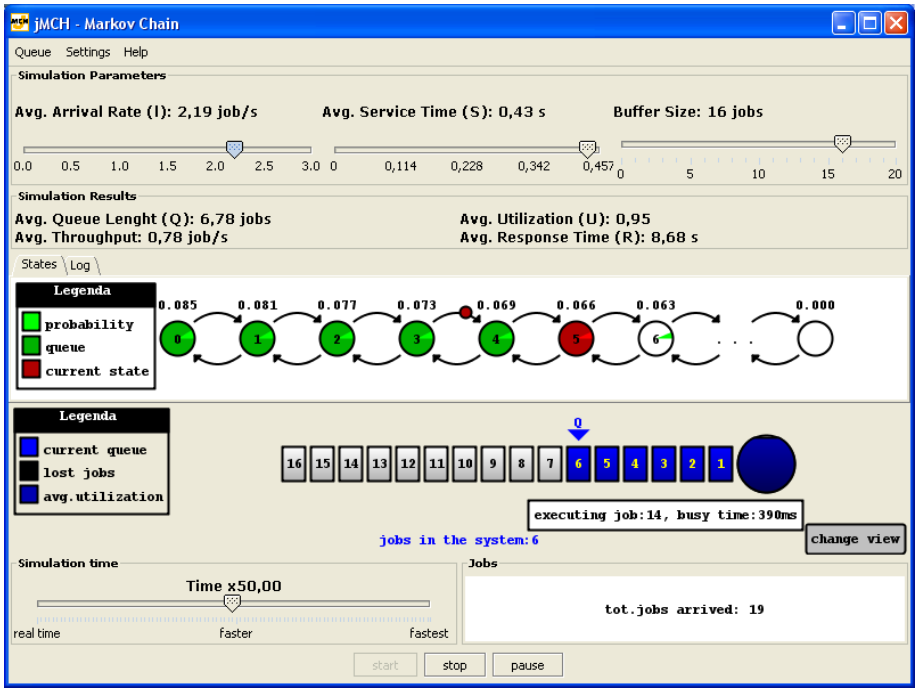



```
Generated by: lqns, version 5.6
Copyright the Real-Time and Distributed Systems Group,
Department of Systems and Computer Engineering
Carleton University, Ottawa, Ontario, Canada. K1S 5B6

Invoked as: lqns .\examples\ex1-1.lqnx
Input:  .\examples\ex1-1.lqnx
Comment: Example Layered Queueing Network

Convergence test value: 2.15474e-006
Task Name       Entry Name      Phase 1     Phase 2     coeff of var **2
c1              c1              0           232.254     1.35729
c2              c2              0           685.33      1.26521
c3              c3              0           331.489     1.33786
c4              c4              0           903.178     1.0475
c5              c5              0           1612.63     0.825956
s1              s1              2.35984     0.25        0.713671
s2              put             0.01        0.64        0.802469
                get             0.81        0.01        0.82


Throughputs and utilizations per phase:

Task Name       Entry Name      Throughput  Phase 1     Phase 2     Total
c1              c1              0.076446    0           1           1
c2              c2              0.0429666   0           1           1
c3              c3              0.0635288   0           1           1
c4              c4              0.0340557   0           1           1
c5              c5              0.0226314   0           1           1
s1              s1              0.57265     0.808758    0.286325    1.09508
s2              put             0.308704    0.0308704   0.246963    0.277834
                get             0.213457    0.192111    0.0213457   0.213457
Total:                          0.522161    0.222982    0.268309    0.491291


Utilization and waiting per phase for processor:  c1

Task Name       Pri n   Entry Name      Utilization Phase 1     Phase 2
c1              0   1   c1              0.535122    0           0


Utilization and waiting per phase for processor:  c2

Task Name       Pri n   Entry Name      Utilization Phase 1     Phase 2
c2              0   1   c2              0.300766    0           0
```

Software models vs Performance indices

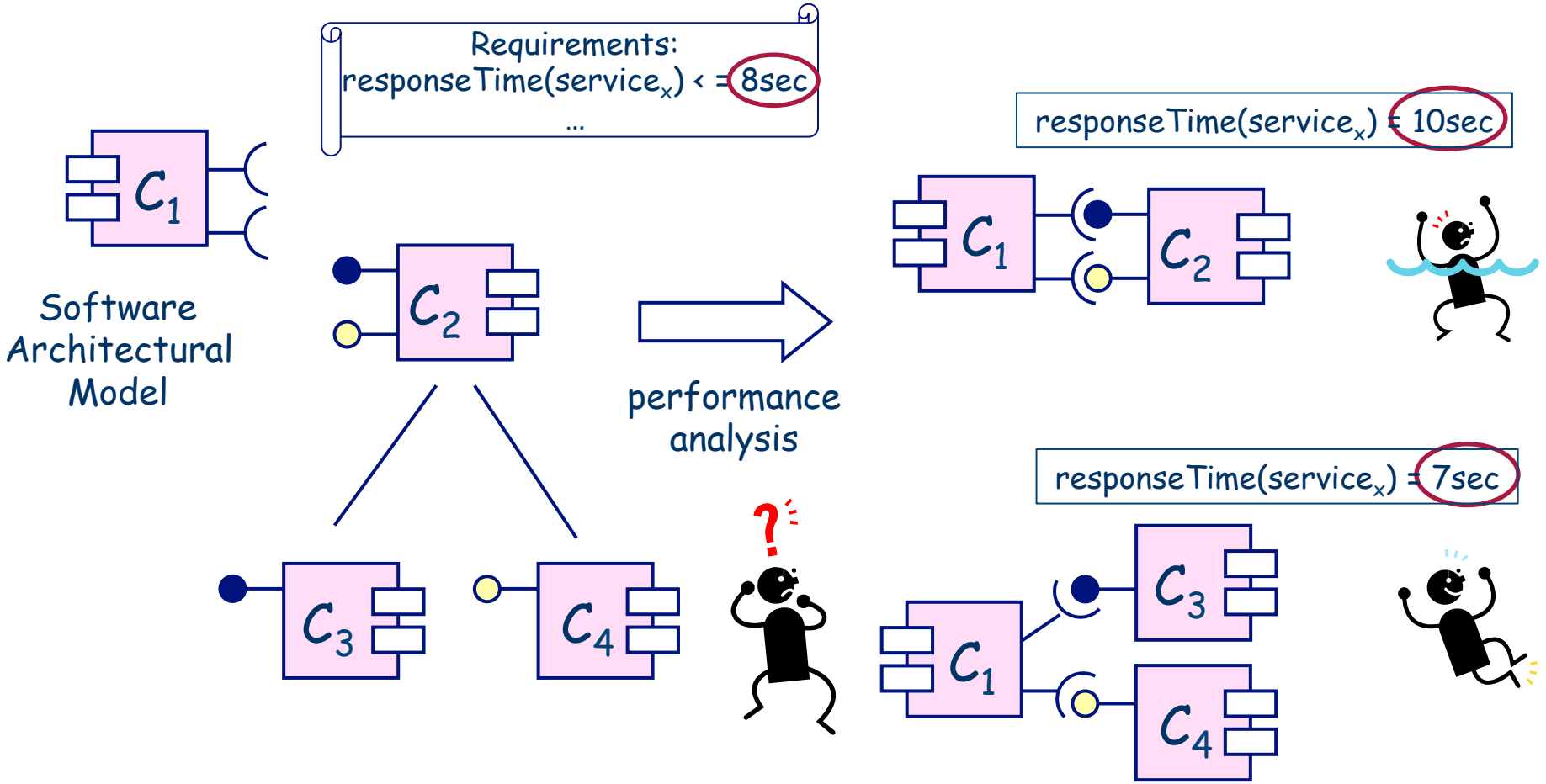
The interpretation of performance indices is not a trivial task!





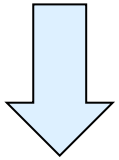
Software models vs Performance indices

What to change in order to improve the software performance?



Software models vs Performance indices

What to change in order to improve the software performance?



- **Structure** (e.g. splitting components)
- **Behavior** (e.g. reducing number of messages)
- **Deployment** (e.g. moving components across hosts)

Models are here intended as **instruments to support decisions** along the software lifecycle

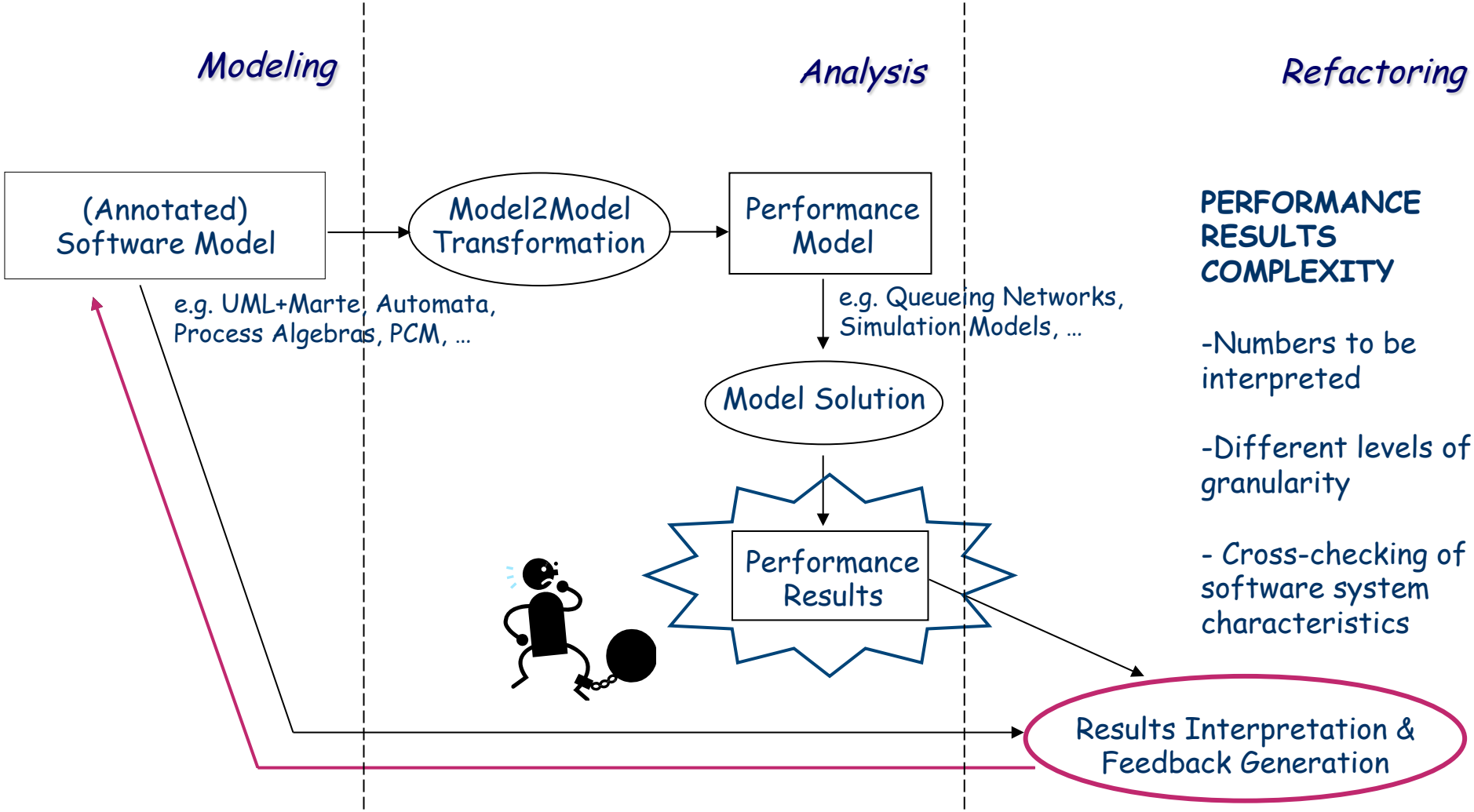
Outline

- Problem statement
- (Logic-based) Reasoning on Performance Antipatterns
- Performance Antipatterns in Modeling Languages
  - Unified Modeling Language (UML)
  - A Domain Specific Language (PCM)
  - An Architecture Description Language (AEmilia)
- Advanced MDE techniques

PROBLEM STATEMENT…

… AND DIFFERENT WAYS TO APPROACH IT

» Software performance process

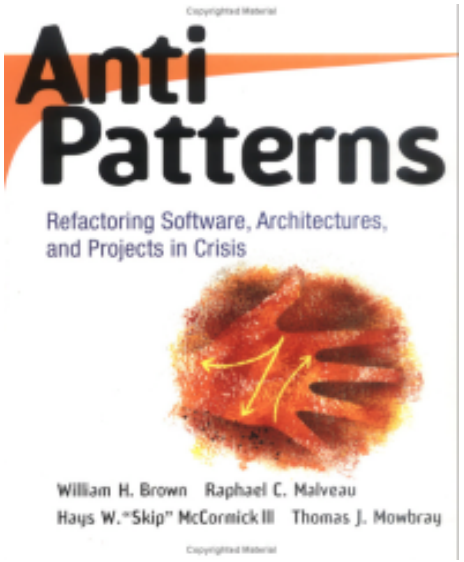


» Software Performance Feedback: state-of-the-art

		Approach	(Annotated) Software Architectural Model	Performance Model	Framework
Antipattern-based		Williams et al. 2002	Software Execution Model	System Execution Model	SPE-ED
		……	……	……	……
		Parsons et al. 2008	JEE systems	Reconstructed runtime model	PAD
Rule-based		Barber et al. 2002	Domain Reference Arch.	Simulation Model	RARE /ARCADE
		……	……	……	……
		Xu 2010	UML	Layered QN	Perf. Booster
Design Space Exploration	Simple Criteria	Zheng et al. 2003	UML	Simulation Model	-
		……	……	……	……
		Ipek et al. 2008	Artificial Neural Network	Simulation Model	-
	Meta-heuristics	Canfora et al. 2005	Workflow Model	Workflow QoS Model	-
		……	……	……	……
		Martens et al. 2010	PCM	Simulation Model	PerOpteryx

» Antipattern-based approaches

- They make use of antipatterns knowledge to cope with performance issues



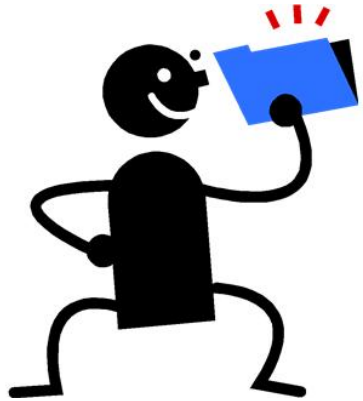
Williams, L.G., Smith, C.U.: PASA(SM): An Architectural Approach to Fixing Software Performance Problems. In: International Computer Measurement Group Conference, Computer Measurement Group (2002)

...

Parsons, T., Murphy, J.: Detecting Performance Antipatterns in Component Based Enterprise Systems. Journal of Object Technology 7 (2008)

» Rule-based approaches

- They encapsulate general knowledge on how to improve system performance into executable rules



Barber, K.S., Graser, T.J., Holt, J.: Enabling Iterative Software Architecture Derivation Using Early Non-Functional Property Evaluation. In: ASE, IEEE Computer Society (2002)

...

Xu, J.: Rule-based automatic software performance diagnosis and improvement. Perform. Eval. 67 (2010)

» Design space exploration – simple criteria

- They explore the design space by

  examining alternatives that can

  cope with performance flaws

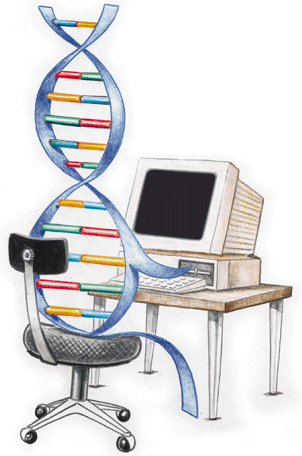
Zheng, T., Woodside, C.M.: Heuristic optimization of scheduling and allocation for distributed systems with soft deadlines. In Computer Performance Evaluation /TOOLS (2003)

…

Ipek, E., McKee, S.A., Singh, K., Caruana, R., de Supinski, B.R., Schulz, M.: Efficient architectural design space exploration via predictive modeling. ACM Trans. on Architecture and Code Optimization (2008)

» Design space exploration – Metaheuristics

- They make use of evolutionary algorithms that look for design alternatives aimed at improving the system performance



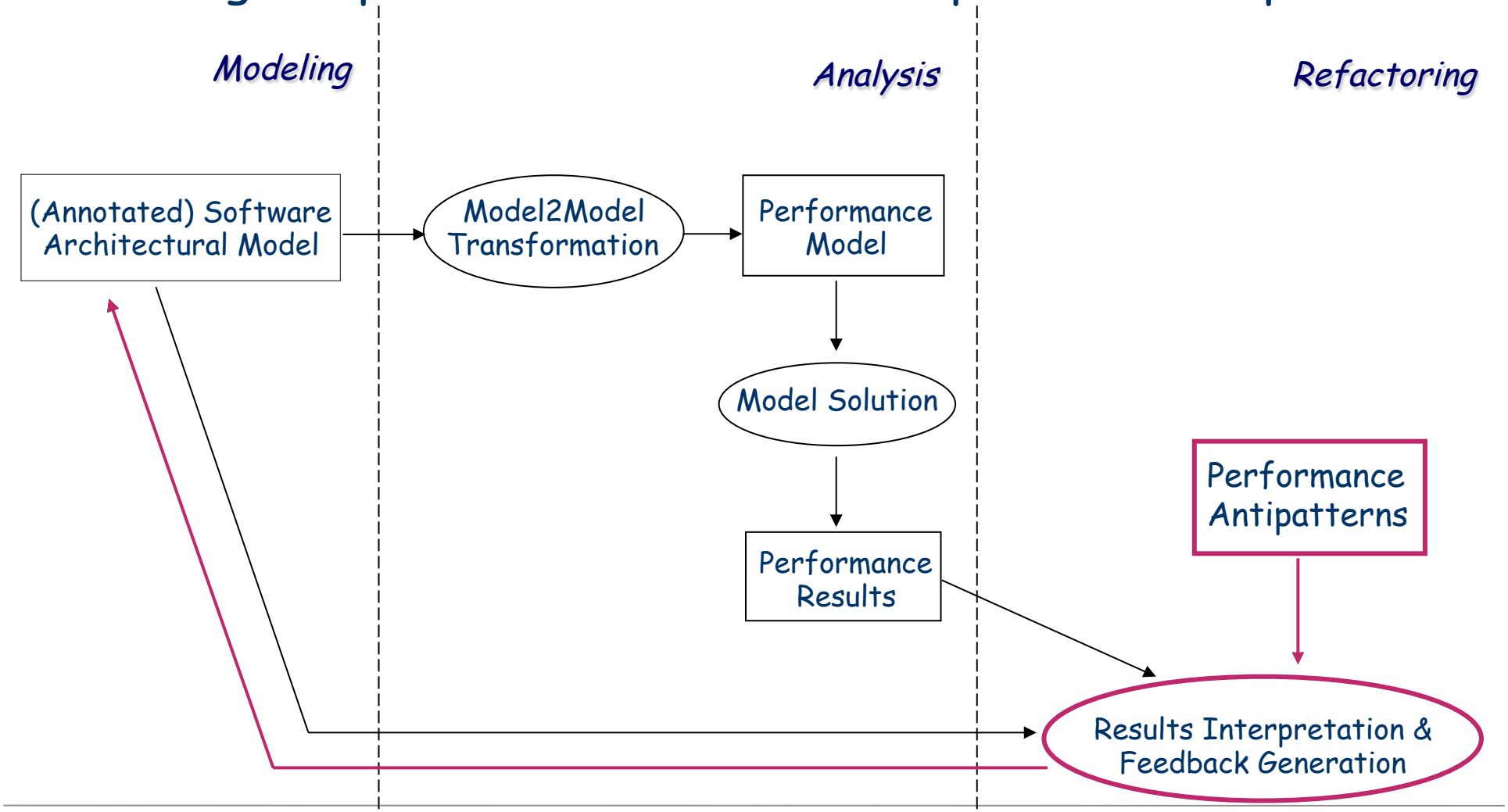
Canfora, G., Penta, M.D., Esposito, R., Villani, M.L.: An approach for QoS-aware service composition based on genetic algorithms. In Beyer, H.G., O'Reilly, U.M., eds.: GECCO, ACM (2005)

...

Martens, A., Koziolek, H., Becker, S., Reussner, R.: Automatically improve software architecture models for performance, reliability, and cost using evolutionary algorithms. In WOSP/SIPEW (2010)

	Approach	(Annotated) Software Architectural Model	Performance Model	Framework
Antipatterns-based	Williams et al. 2002	Software Execution Model	System Execution Model	SPE-ED
	……	……	……	……
	Parsons et al. 2008	JEE systems	Reconstructed runtime model	PAD
	Focus of this course	Unified Modeling Language (UML), Palladio Component Model (PCM), AEmilia ADL	Queueing Network, Simulation Model, Markov Chain	Performance Antipatterns aNd feeDback on software Architectures (PANDA)
Rule-based	Barber et al. 2002	Domain Reference Arch.	Simulation Model	RARE /ARCADE
	……	……	……	……
	Xu 2010	UML	Layered QN	Perf. Booster
…	……			
	……			

» Adding antipatterns in the software performance process



» **Antipatterns**: Negative features of a software system

» Conceptually similar to design patterns: recurring solutions to common design problems

» The definition includes common mistakes (i.e. bad practices) in software development as well as their solutions

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

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

» Performance Antipatterns : why are they complex?

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.
...	...	...

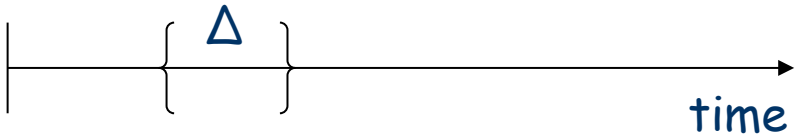
Performance Antipatterns classification

	Antipattern		Problem	Solution
Single-value	Unbalanced Processing	Concurrent Processing Systems	Processing cannot make use of available processors.	Restructure software or change scheduling algorithms to enable concurrent execution.
		"Pipe and Filter" Architectures	The slowest filter in a "pipe and filter" architecture causes the system to have unacceptable throughput.	Break large filters into more stages and combine very small ones to reduce overhead.
		Extensive Processing	Extensive processing in general impedes overall response time.	Move extensive processing so that it doesn't impede high traffic or more important work.
	...		...	...
Multiple-values	The Ramp		Occurs when processing time increases as the system is used.	Select algorithms or data structures based on maximum size or use algorithms that adapt to the size.

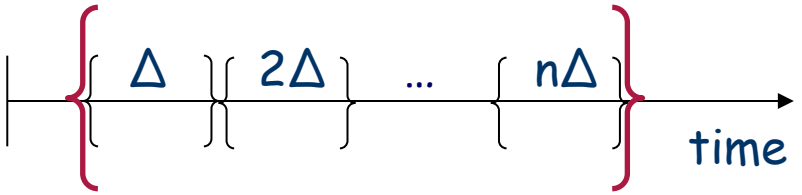
C. U. Smith and L. G.Williams. "More new software performance antipatterns: Even more ways to shoot yourself in the foot", 2003.

Single-value vs Multiple-values

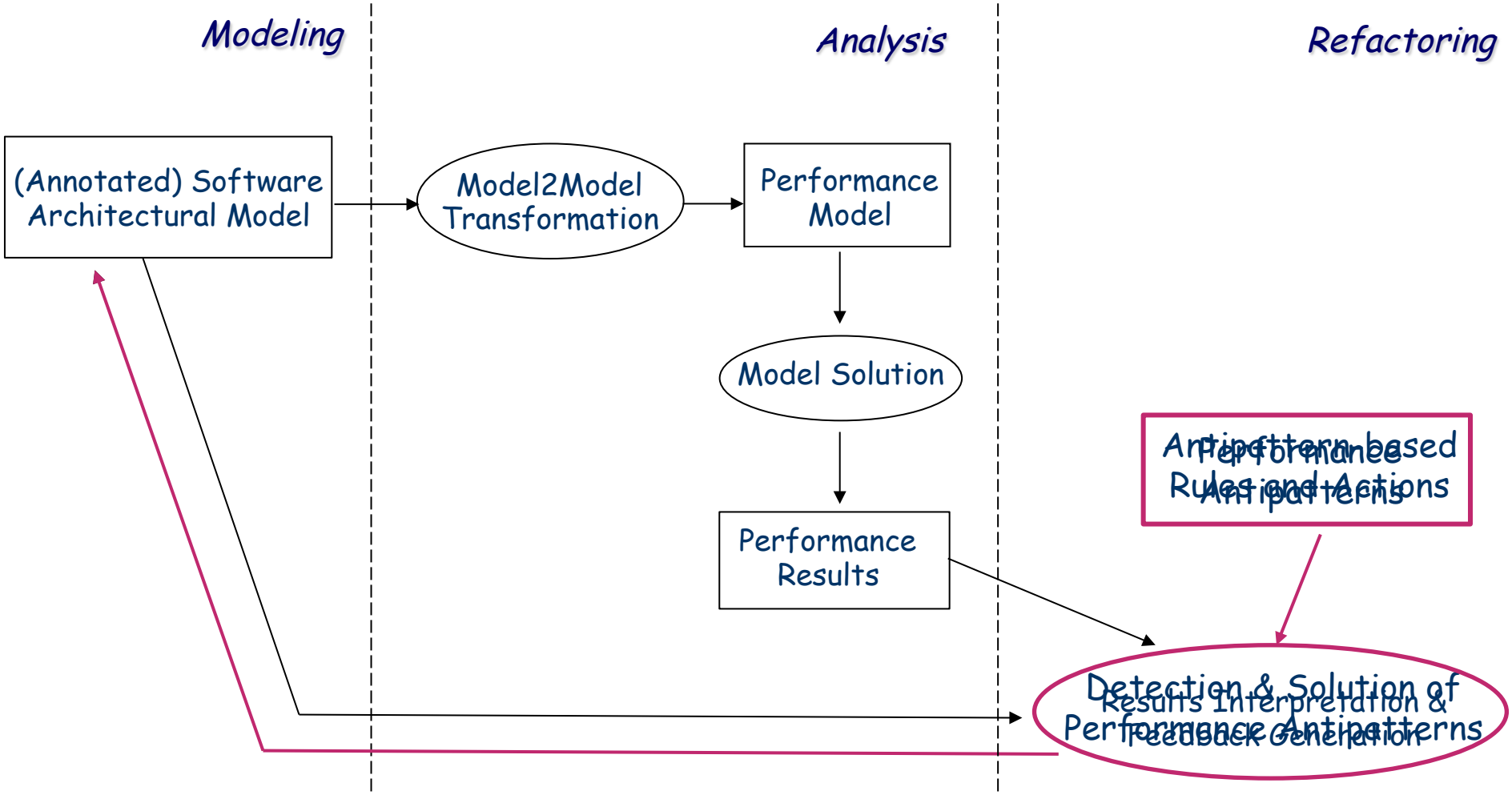
» Single-value: performance indices are evaluated in a certain interval, i.e. the mean, max or min values.



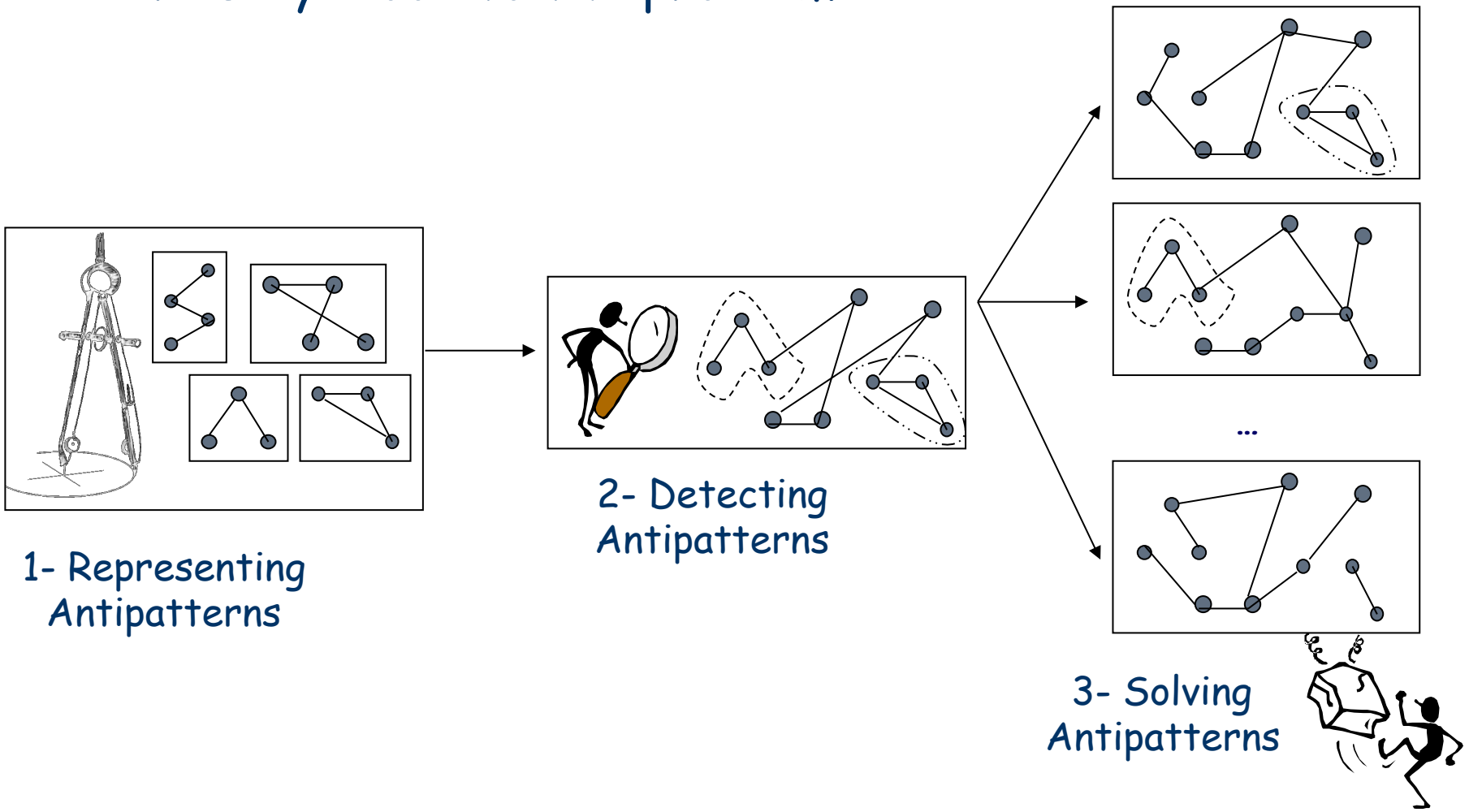
» Multiple-values: performance indices are evaluated along the time, i.e. the values trend (or evolution).



» Software performance process: introducing **automation**

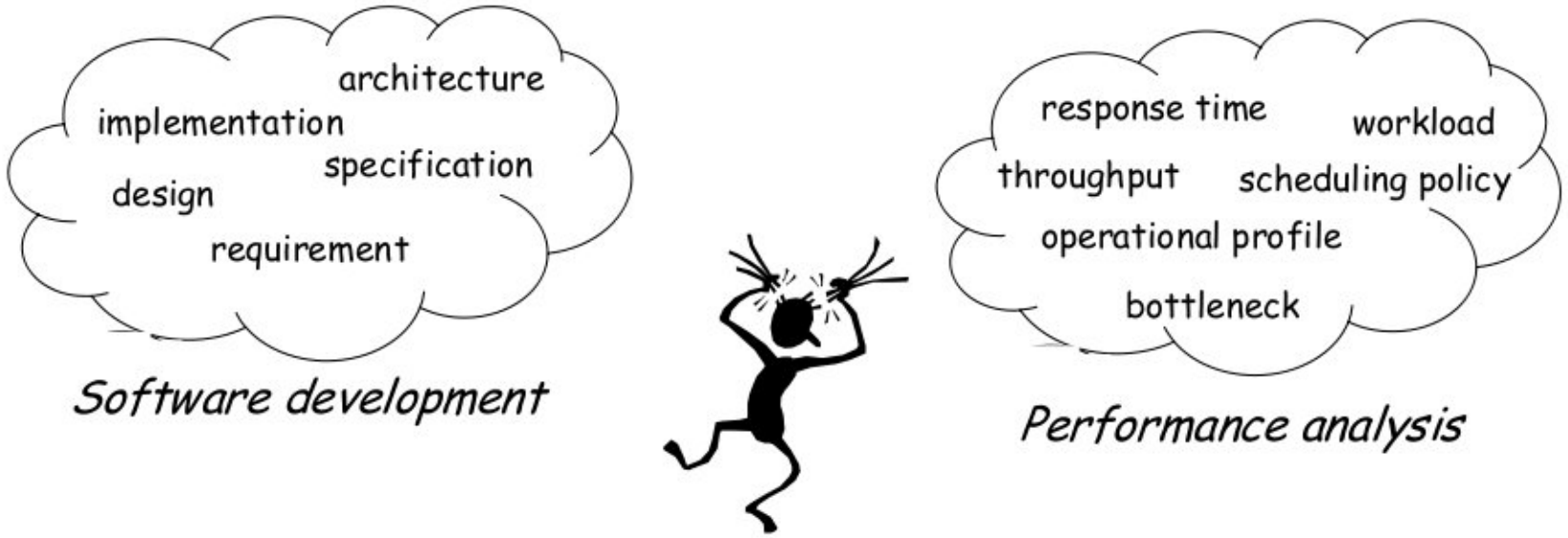


» A bird's-eye look to the problem

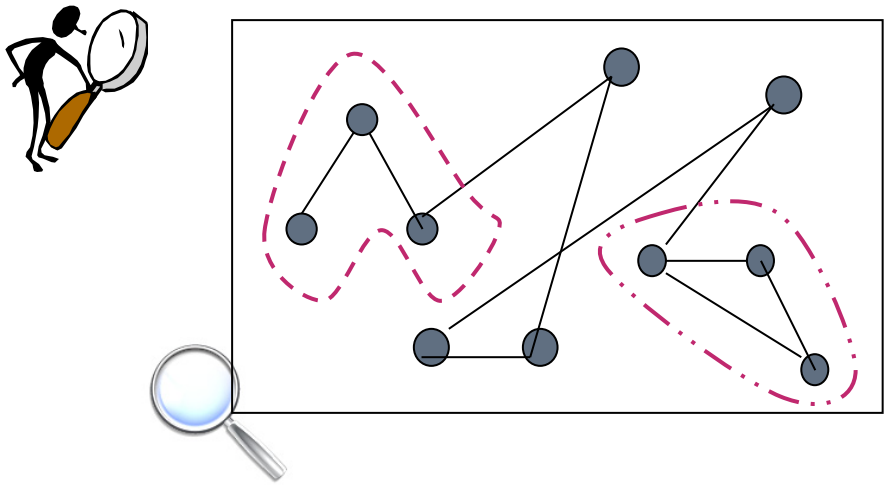


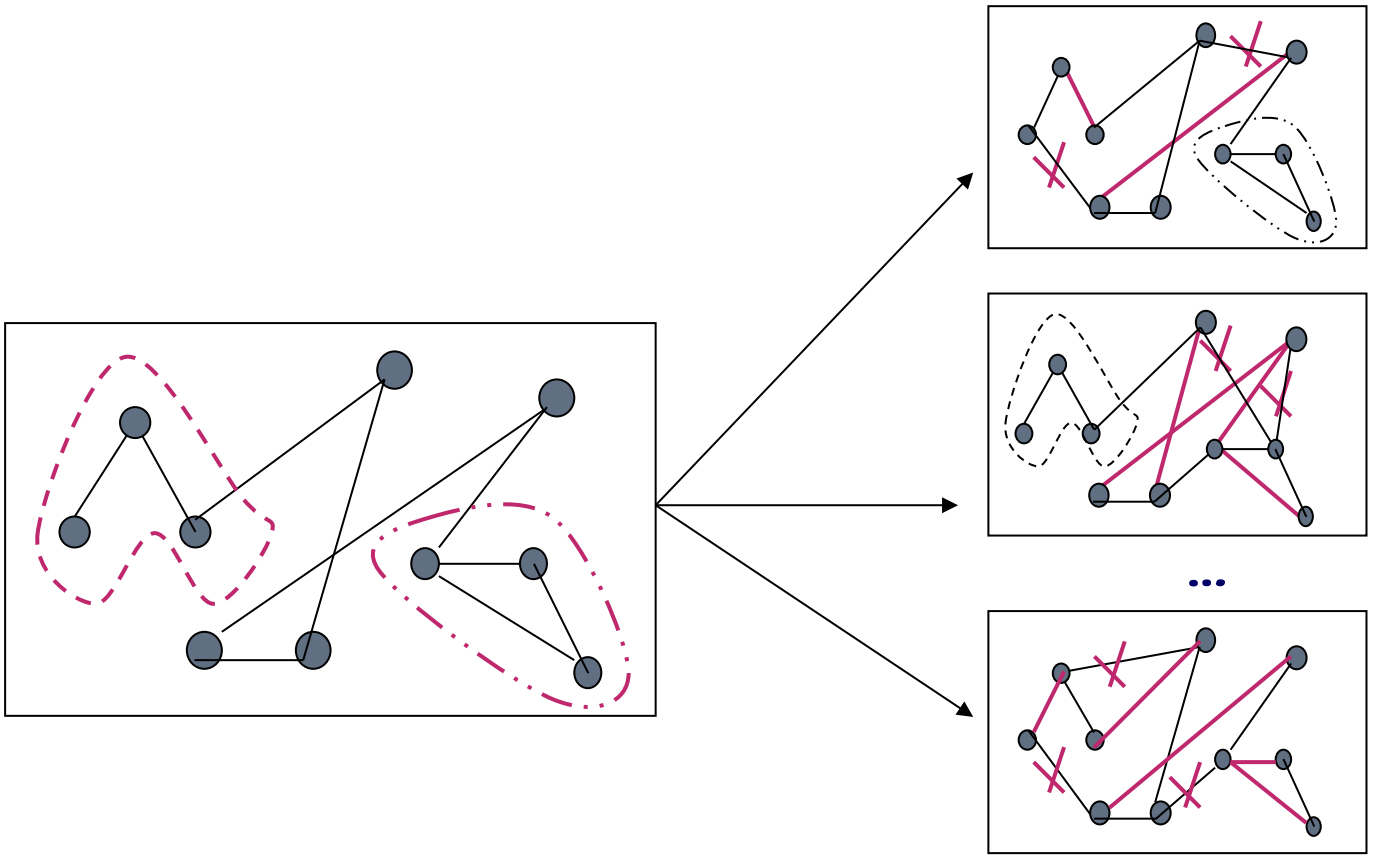
Representing Antipatterns:

What are the software architectural model elements we need for representing antipatterns?



Detecting Antipatterns:
How to explore the architectural models to recognize antipattern occurrences?





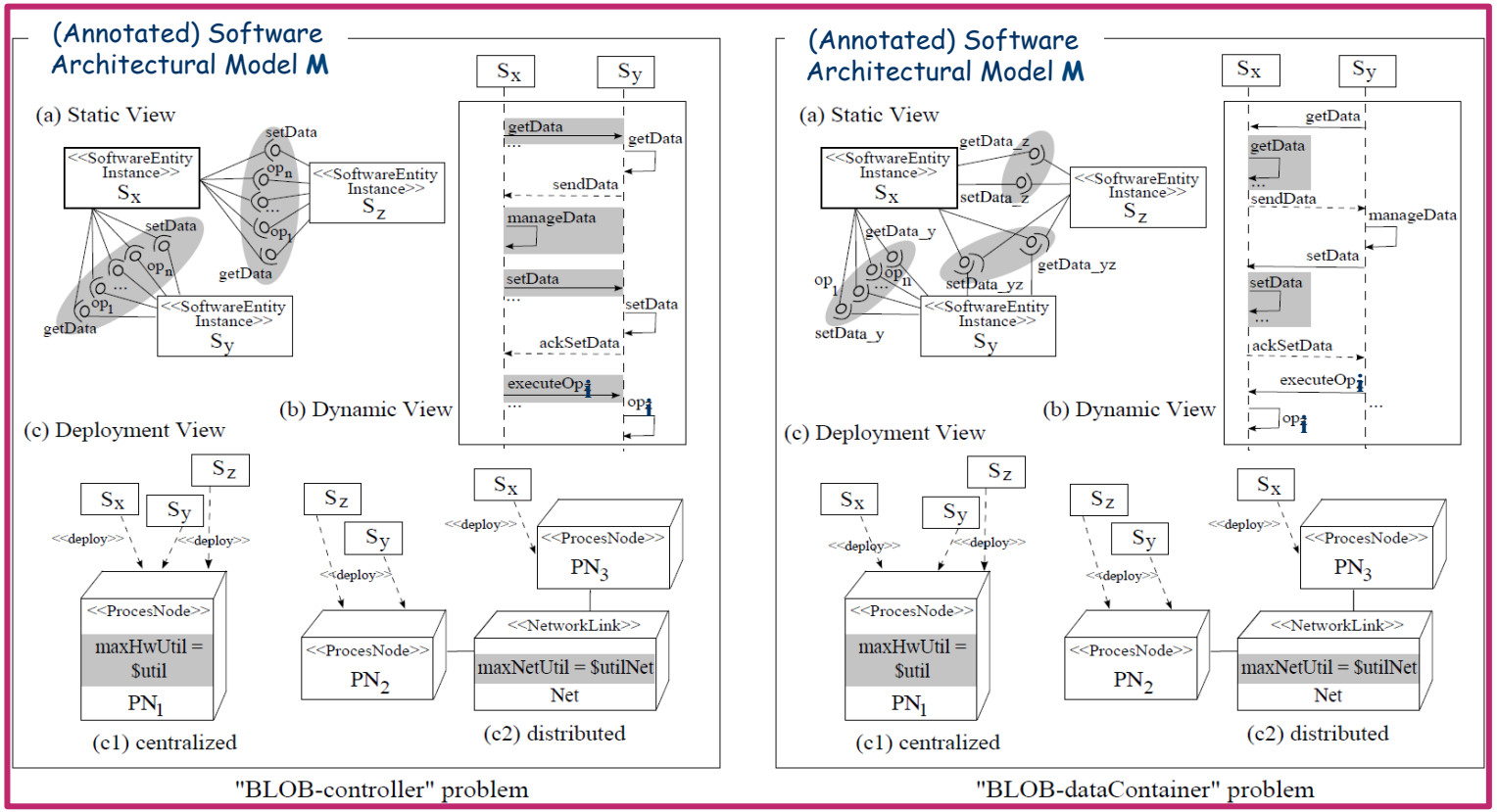
Solving Antipatterns:

What are the ***refactoring actions*** that lead the architectural model to remove performance flaws?

(LOGIC-BASED) REASONING ON PERFORMANCE ANTIPATTERNS

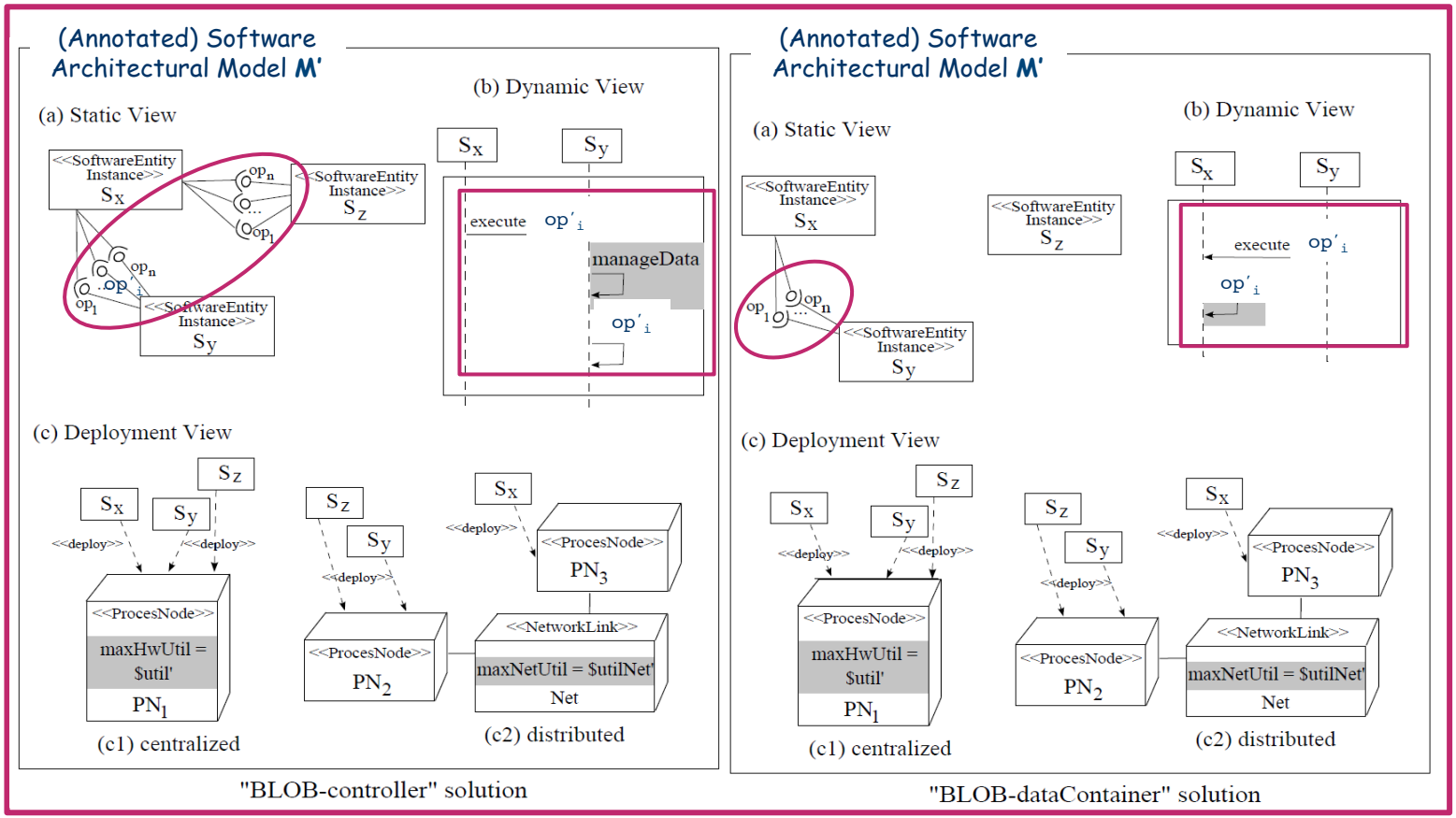
» Graphical representation of the "Blob" Antipattern

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*"



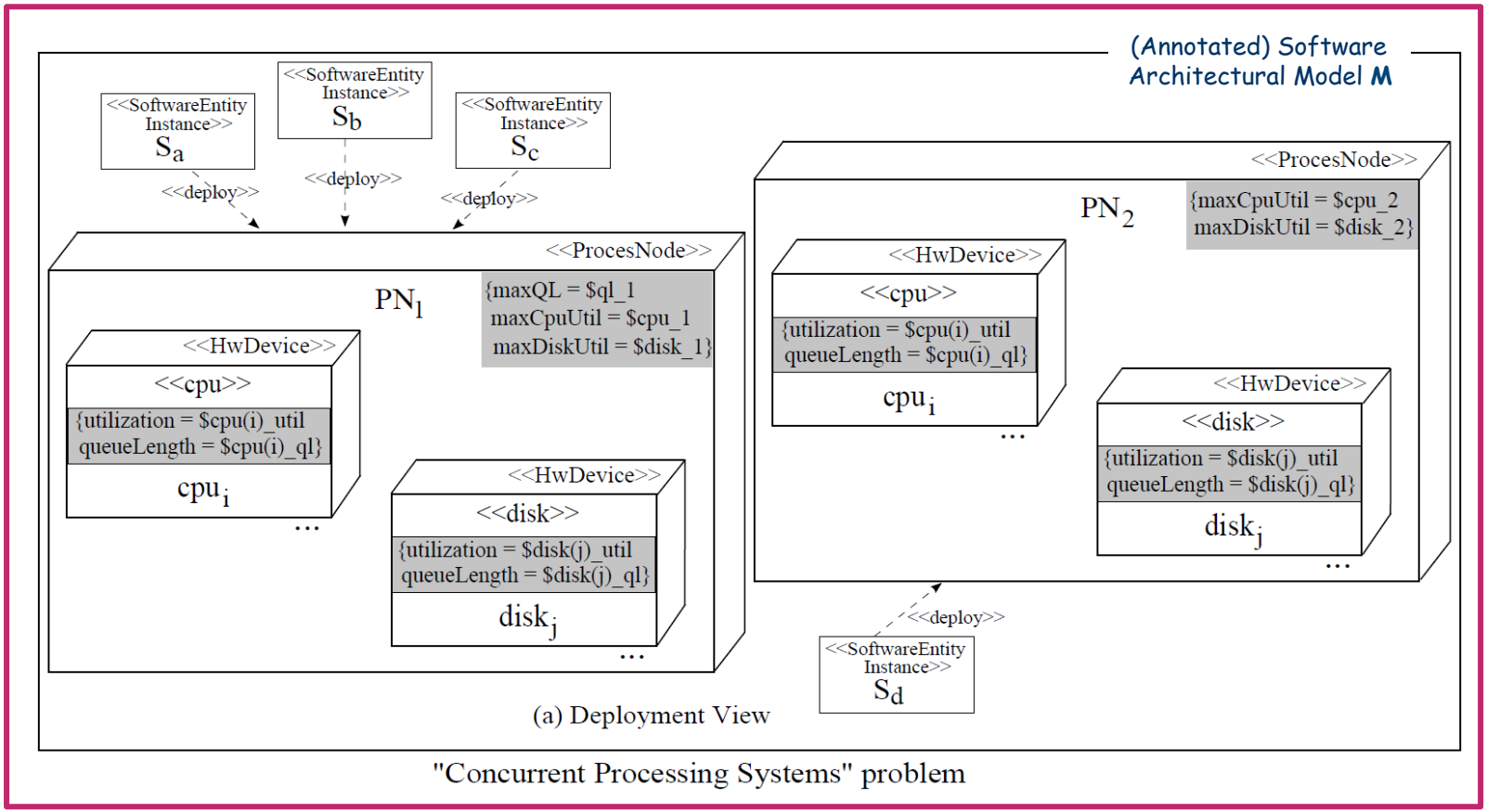
» Graphical representation of the "Blob" Antipattern

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



» Graphical representation of the "Concurrent Processing Systems" Antipattern

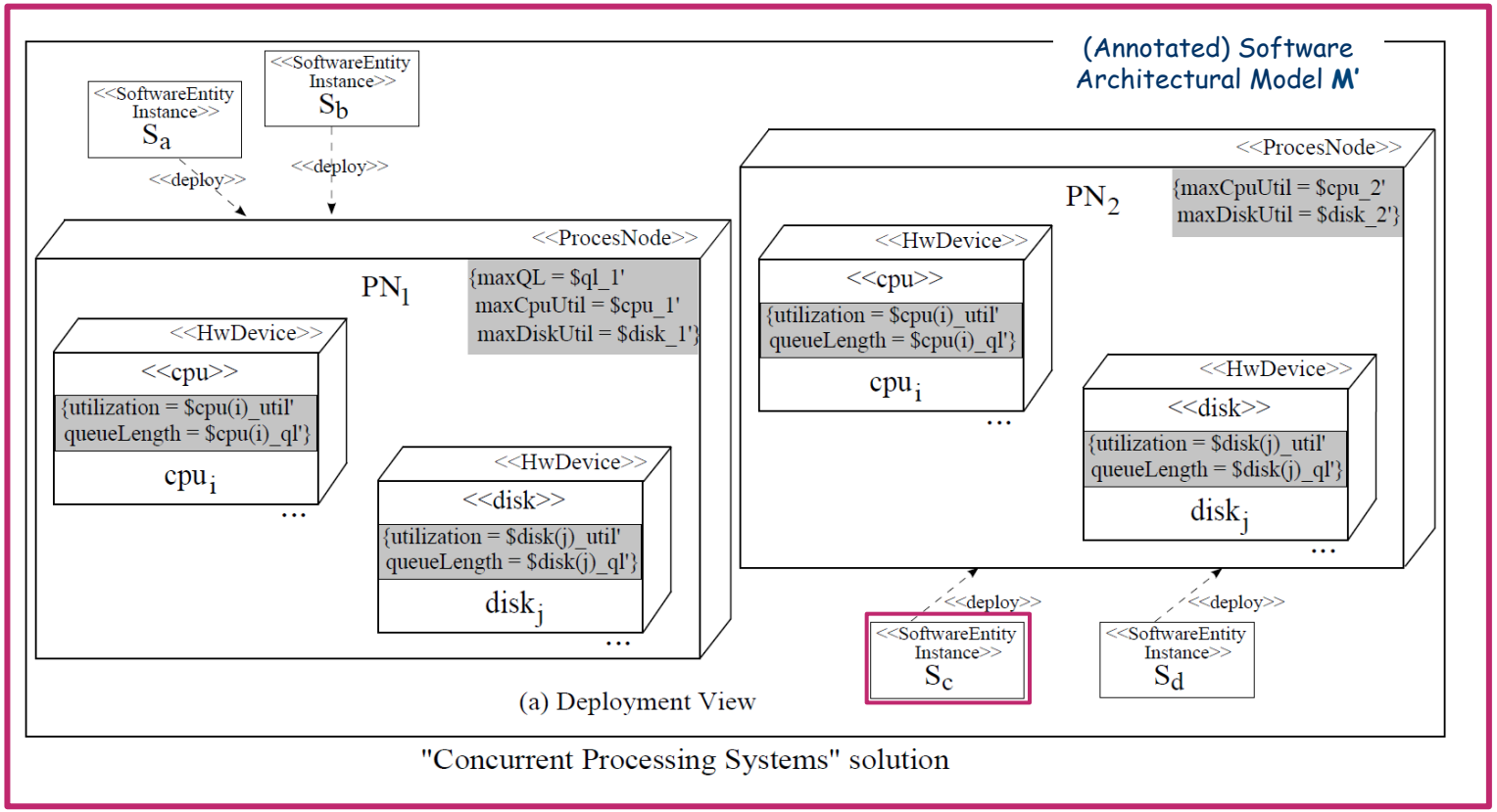
PROBLEM: "Occurs when processing cannot make use of available processors"



"Concurrent Processing Systems" problem

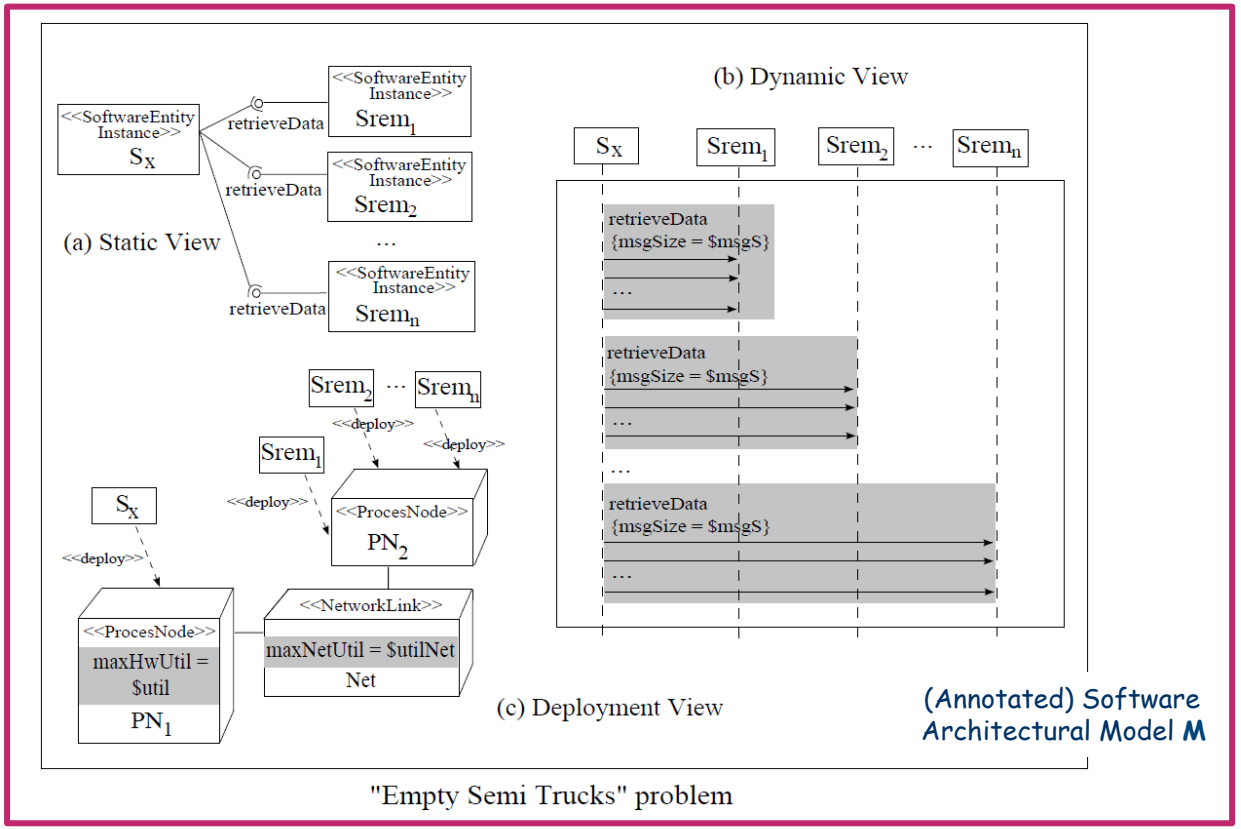
» Graphical representation of the "Concurrent Processing Systems" Antipattern

SOLUTION: *"Restructure software or change scheduling algorithms to enable concurrent execution"*



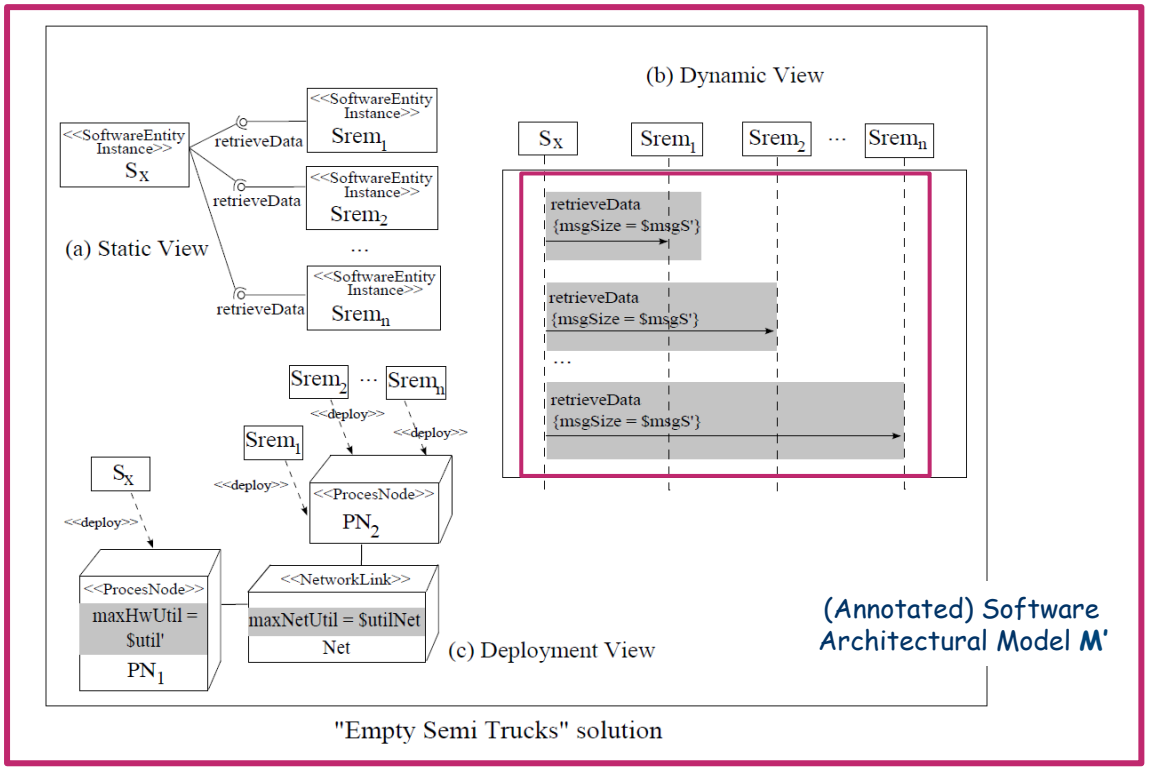
» Graphical representation of the"Empty Semi Trucks" Antipattern

PROBLEM: "*Occurs when an excessive number of requests is required to perform a task. It may be due to inefficient interface that, in some cases, implies an inefficient use of available bandwidth (low size messages)*"



» Graphical representation of the "Empty Semi Trucks" Antipattern

SOLUTION: "***The Batching performance pattern combines items into messages to make better use of available bandwidth (in case of low size messages).***
The Coupling performance pattern, Session Facade design pattern, and Aggregate Entity design pattern provide more efficient interfaces."



"Empty Semi Trucks" solution

First Order Logic-based representation of antipatterns

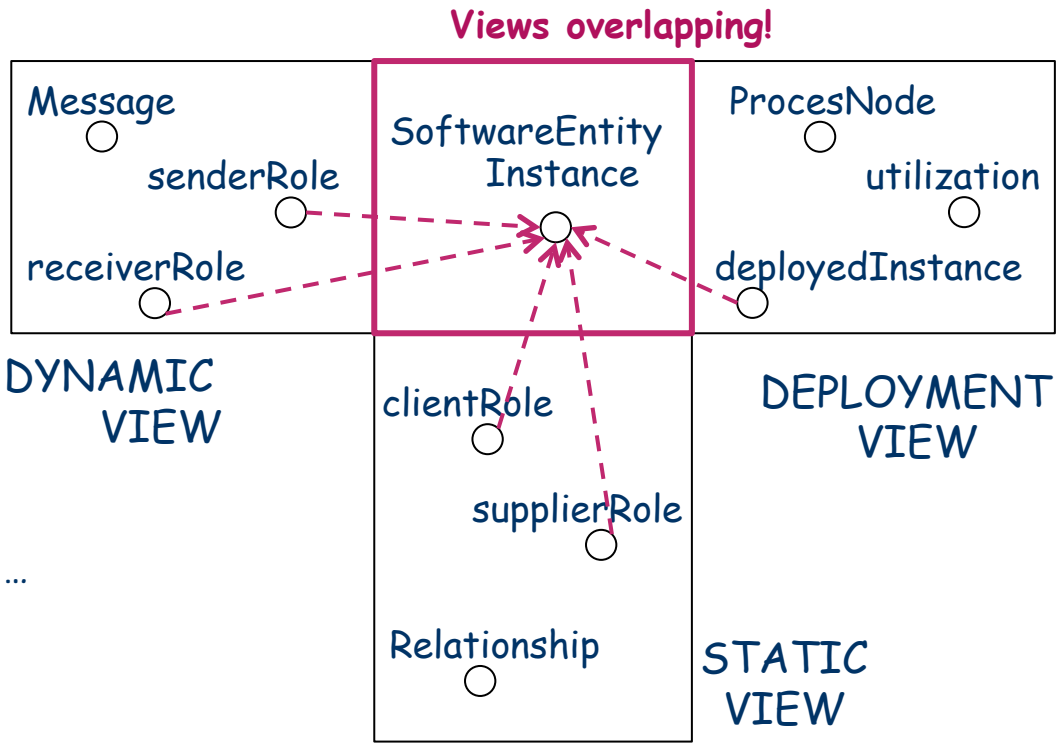
How to make more "formal" (i.e. machine-processable) the specifications of antipatterns???

» Different model elements of antipatterns can be organized into "Views"

STATIC VIEW: software resources, relationships among them, … to model static aspects

DYNAMIC VIEW: interactions such as messages between sw resources, … to model dynamic aspects

DEPLOYMENT VIEW: hardware resources, … to model deployment aspects

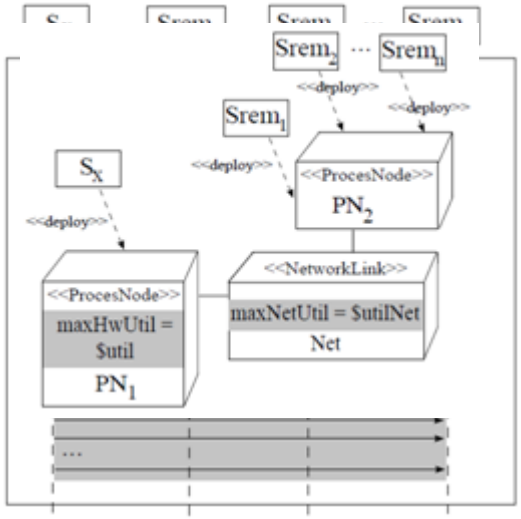


» Basic Idea: a performance antipattern can be formalized as a logical predicate $LP_{antipatName}$

» A logical predicate is made of (Static, Dynamic, Deployment) basic predicates, BP_i

$$LP_{antipatName} = BP_1 (\wedge, \vee) \ \ldots (\wedge, \vee) BP_n$$

» The Empty Semi Trucks antipattern occurs when the following predicate is true:



$$\exists swE_x \in \mathrm{swE}, S \in \mathbb{S} \mid$$
$$F_{numRemMsgs}(swE_x, S) \geq Th_{maxRemMsgs} \wedge$$
$$(F_{maxNetUtil}(P_{swE_x}, swE_x) < Th_{minNetUtil} \vee$$
$$F_{numRemInst}(swE_x, S) \geq Th_{maxRemInst})$$

where swE represents the set of *SoftwareEntityInstance*(s), and $\mathbb{S}$ represents the set of *Service*(s).

All the (swE_x, S) instances satisfying the predicate must be pointed out to the designer for a deeper analysis.

Auxiliary elements in the formalization process

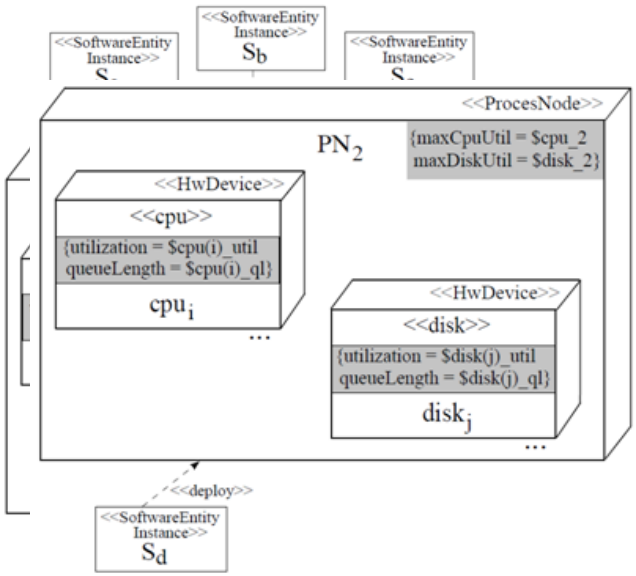
- **Functions**
  - > $F_{functionName}$ elaborates information of the model

  (e.g. $F_{numMsgs}$ is a function counting the number of messages sent by an instance of the class/component model element)

- **Thresholds**
  - > $Th_{thresholdName}$ is a value (*estimated by the software architects*) used to establish the *acceptable range* of values for system features

  (e.g. $Th_{maxMsgs}$ is a threshold value representing the upper bound for an acceptable number of messages exchanged among two software instances. It can be estimated, for example, as the average number of all messages sent by all software entities, plus the corresponding variance)

» The Concurrent Processing Systems antipattern occurs when the following predicate is true:



$$\exists P_x, P_y \in \mathbb{P} \mid$$

$$F_{maxQL}(P_x) \geq Th_{maxQL} \ \wedge$$

$$(\ F_{maxHwUtil}(P_x, cpu) \geq Th_{maxCpuUtil} \ \wedge$$

$$F_{maxHwUtil}(P_x, disk) \geq Th_{maxDiskUtil} \ ) \ \vee$$

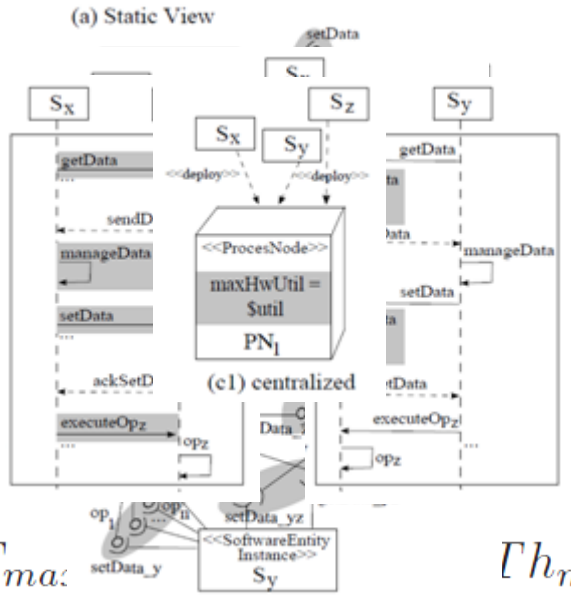
$$(\ F_{maxHwUtil}(P_y, cpu) < Th_{minCpuUtil} \ \wedge$$

$$F_{maxHwUtil}(P_y, disk) < Th_{minDiskUtil} \ )$$

where $\mathbb{P}$ represents the set of *ProcesNode*(s).

All the (P_x, P_y) instances satisfying the predicate must be pointed out to the designer for a deeper analysis.

» The Blob antipattern occurs when the following predicate is true:



$$\exists swE_x, swE_y \in \mathbb{SWE}, S \in \mathbb{S} \mid$$

$$(F_{numClientConnects}(swE_x) \geq Th_{maxConnects} \bigvee$$

$$F_{numSupplierConnects}(swE_x) \geq Th_{maxConnects}) \bigwedge$$

$$(F_{numMsgs}(swE_x, swE_y, S) \geq Th_{maxMsgs} \bigvee$$

$$F_{numMsgs}(swE_y, swE_x, S) \geq Th_{maxMsgs}) \bigwedge$$

$$(F_{ma} \quad Th_{maxHwUtil} \bigvee F_{maxNetUtil}(P \quad _x, P \quad _y) \geq Th_{maxNetUtil})$$

where $\mathbb{SWE}$ represents the set of *SoftwareEntityInstance*(s), and $\mathbb{S}$ represents the set of *Service*(s).

All the (swE_x, swE_y, S) instances satisfying the predicate must be pointed out to the designer for a deeper analysis.

- Each antipattern can be expressed by means of first-order logics
- But this is only OUR interpretation of their textual description

V. Cortellessa, A. Di Marco, and C. Trubiani. "An approach for modeling and detecting Software Performance Antipatterns based on first-order logics", accepted for publication in the journal of Software and Systems Modeling (SoSyM), 2012.

	Antipattern		Formula
Single-value	Blob (or god class/component)		$\exists swE_x, swE_y \in \mathbb{swE}, S \in \mathbb{S} \mid (F_{numClientConnects}(swE_x) \geq Th_{maxConnects} \vee F_{numSupplierConnects}(swE_x) \geq Th_{maxConnects}) \wedge (F_{numMsgs}(swE_x, swE_y, S) \geq Th_{maxMsgs} \vee F_{numMsgs}(swE_y, swE_x, S) \geq Th_{maxMsgs}) \wedge (F_{maxHwUtil}(P_{xy}, all) \geq Th_{maxHwUtil} \vee F_{maxNetUtil}(P_{swE_x}, P_{swE_y}) \geq Th_{maxNetUtil})$
	Unbalanced Processing	Concurrent Processing Systems	$\exists P_x, P_y \in \mathbb{P} \mid F_{maxQL}(P_x) \geq Th_{maxQL} \wedge [(F_{maxHwUtil}(P_x, cpu) \geq Th_{maxCpuUtil} \wedge F_{maxHwUtil}(P_y, cpu) < Th_{minCpuUtil}) \vee (F_{maxHwUtil}(P_x, disk) \geq Th_{maxDiskUtil} \wedge (F_{maxHwUtil}(P_y, disk) < Th_{minDiskUtil}))]$
		"Pipe and Filter" Architectures	$\exists OpI \in \mathbb{O}, S \in \mathbb{S} \mid \forall i : F_{resDemand}(Op)[i] \geq Th_{resDemand}[i] \wedge F_{probExec}(S, OpI) = 1 \wedge (F_{maxHwUtil}(P_{swE_x}, all) \geq Th_{maxHwUtil} \vee F_{RT}(S) < Th_{SthReq})$
		Extensive Processing	$\exists OpI_1, OpI_2 \in \mathbb{O}, S \in \mathbb{S} \mid \forall i : F_{resDemand}(Op_1)[i] \geq Th_{maxOpResDemand}[i] \wedge \forall i : F_{resDemand}(Op_2)[i] < Th_{minOpResDemand}[i] \wedge F_{probExec}(S, OpI_1) + F_{probExec}(S, OpI_2) = 1 \wedge (F_{maxHwUtil}(P_{swE_x}, all) \geq Th_{maxHwUtil} \vee F_{RT}(S) > Th_{SrtReq})$
	Circuitous Treasure Hunt		$\exists swE_x, swE_y \in \mathbb{swE}, S \in \mathbb{S} \mid swE_y.isDB = true \wedge F_{numDBmsgs}(swE_x, swE_y, S) \geq Th_{maxDBmsgs} \wedge F_{maxHwUtil}(P_{swE_y}, all) \geq Th_{maxHwUtil} \wedge F_{maxHwUtil}(P_{swE_y}, disk) > F_{maxHwUtil}(P_{swE_y}, cpu)$
	Empty Semi Trucks		$\exists swE_x \in \mathbb{swE}, S \in \mathbb{S} \mid F_{numRcmMsgs}(swE_x, S) \geq Th_{maxRcmMsgs} \wedge F_{maxNetUtil}(P_{swE_x}, swE_x) < Th_{minNetUtil} \vee F_{numRcmInst}(swE_x, S) \geq Th_{maxRcmInst})$
	Tower of Babel		$\exists swE_x \in \mathbb{swE}, S \in \mathbb{S} \mid F_{numExF}(swE_x, S) \geq Th_{maxExF} \vee F_{maxHwUtil}(P_{swE_x}, all) \geq Th_{maxHwUtil}$
	One-Lane Bridge		$\exists swE_x \in \mathbb{swE}, S \in \mathbb{S} \mid F_{numSynchCalls}(swE_x, S) \gg F_{poolSize}(swE_x) \wedge F_{serviceTime}(P_{swE_x}) \ll F_{waitingTime}(P_{swE_x}) \wedge F_{RT}(S) > Th_{SrtReq}$
	Excessive Dynamic Allocation		$\exists S \in \mathbb{S} \mid (F_{numCreatedObj}(S) \geq Th_{maxCrObj} \vee F_{numDestroyedObj}(S) \geq Th_{maxDcObj}) \wedge F_{RT}(S) > Th_{SrtReq}$
Multiple-values	Traffic Jam		$\exists OpI \in \mathbb{O} \mid \frac{\sum_{1<t<k}\lvert(F_{RT}(OpI,t)-F_{RT}(OpI,t-1))\rvert}{k-1} < Th_{OpRtVar} \wedge F_{RT}(OpI,k) - F_{RT}(OpI,k-1) > Th_{OpRtVar} \wedge \frac{\sum_{k<t<n}\lvert(F_{RT}(OpI,t)-F_{RT}(OpI,t-1))\rvert}{n-k} < Th_{OpRtVar}$
	The Ramp		$\exists Op \in \mathbb{O} \mid \frac{\sum_{1<t<n}\lvert(F_{RT}(OpI,t)-F_{RT}(OpI,t-1))\rvert}{n} > Th_{OpRtVar} \wedge \frac{\sum_{1<t<n}\lvert(F_T(OpI,t)-F_T(OpI,t-1))\rvert}{n} > Th_{OpThVar}$
	More is Less		$\exists P_x \in \mathbb{P} \mid \forall i : F_{par}(P_x)[i] \ll \frac{\sum_{1<t<N}(F_{RTpar}(P_x,t)[i]-F_{RTpar}(P_x,t-1)[i])}{N}$

Key-Question:

Once we have (somehow) "represented" antipatterns,

how can we detect them in a software model?

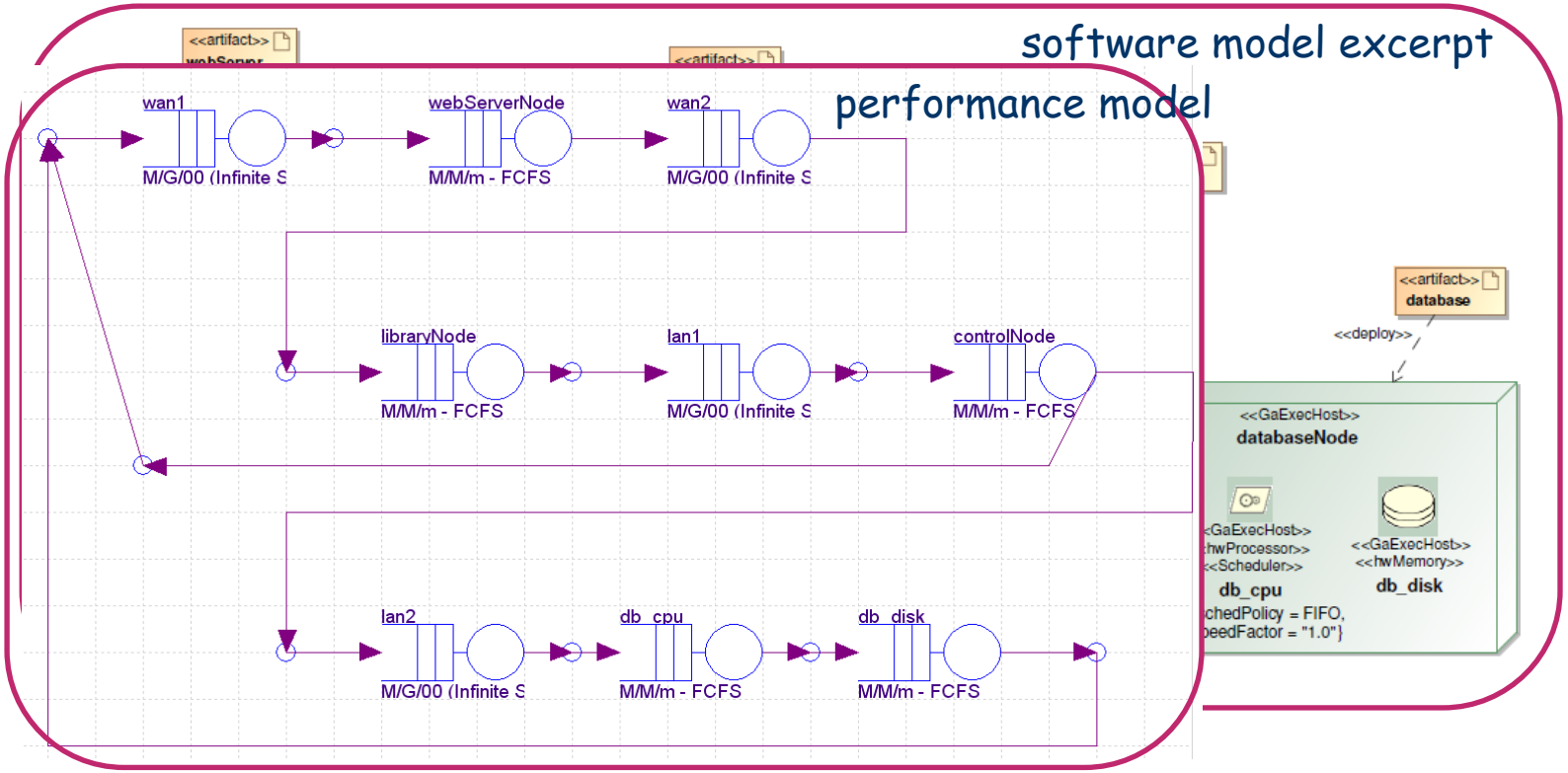
First rough approach

a Java rule-engine application able to parse any XML document compliant to a Schema that we have defined as antipattern vocabulary.

An excerpt of the Java application for the "Concurrent Processing Systems" antipattern

```
boolean checkCPS_PA(Document doc) {
    boolean PA_cps = false;

    int nodeQLs[] = new int[numProcNodes(doc)];
    float nodeHwResUtil[] = new float[numProcNodes(doc)];

    NodeList listOfProcesNodes = doc.getElementsByTagName("procesNode");

    for (int s = 0; s < listOfProcesNodes.getLength(); s++) {
        float maxUtilCurrentNode = 0;
        Node procesNode = listOfProcesNodes.item(s);
        if (procesNode.getNodeType() == Node.ELEMENT_NODE) {
            Element procesElement = (Element) procesNode;

            NodeList innerListOfHwResources = procesElement.
                  getElementsByTagName("hwResource");
            for (int i = 0; i < innerListOfHwResources.getLength(); i++) {
                Node hwResNode = innerListOfHwResources.item(i);
                float tempUtilCurrentNode = Float.parseFloat(hwResNode.
                      getAttributes().item(1).getNodeValue());
                if (tempUtilCurrentNode > maxUtilCurrentNode) {
                    maxUtilCurrentNode = tempUtilCurrentNode;
                }
            }
            nodeHwResUtil[s] = maxUtilCurrentNode;

            NodeList queueLengthList = procesElement.getElementsByTagName("
                  queueLength");
            Element queueLengthElement = (Element) queueLengthList.item(0);
            nodeQLs[s] = Integer.parseInt(((Node) queueLengthElement.
                  getChildNodes().item(0)).getNodeValue().trim());
        }
    }

    for (int i = 0; i < numProcNodes(doc); i++) {
        if (nodeQLs[i] > getTh_queue(doc)) {
            if (nodeHwResUtil[i] > getTh_hwRes(doc)) {
                for (int j = 0; j < numProcNodes(doc); j++) {
                    if (nodeHwResUtil[i] > nodeHwResUtil[j]) {
                        PA_cps = true;
                    }
                }
            }
        }
    }
    return PA_cps;
}
```

» A case study: the "E-commerce System"



Requirement	Required Value	Predicted Value
RT(browseCatalog)	1.2 sec	1.5 sec
RT(makePurchase)	2 sec	2.77 sec

» A preliminary complex step: **setting thresholds**

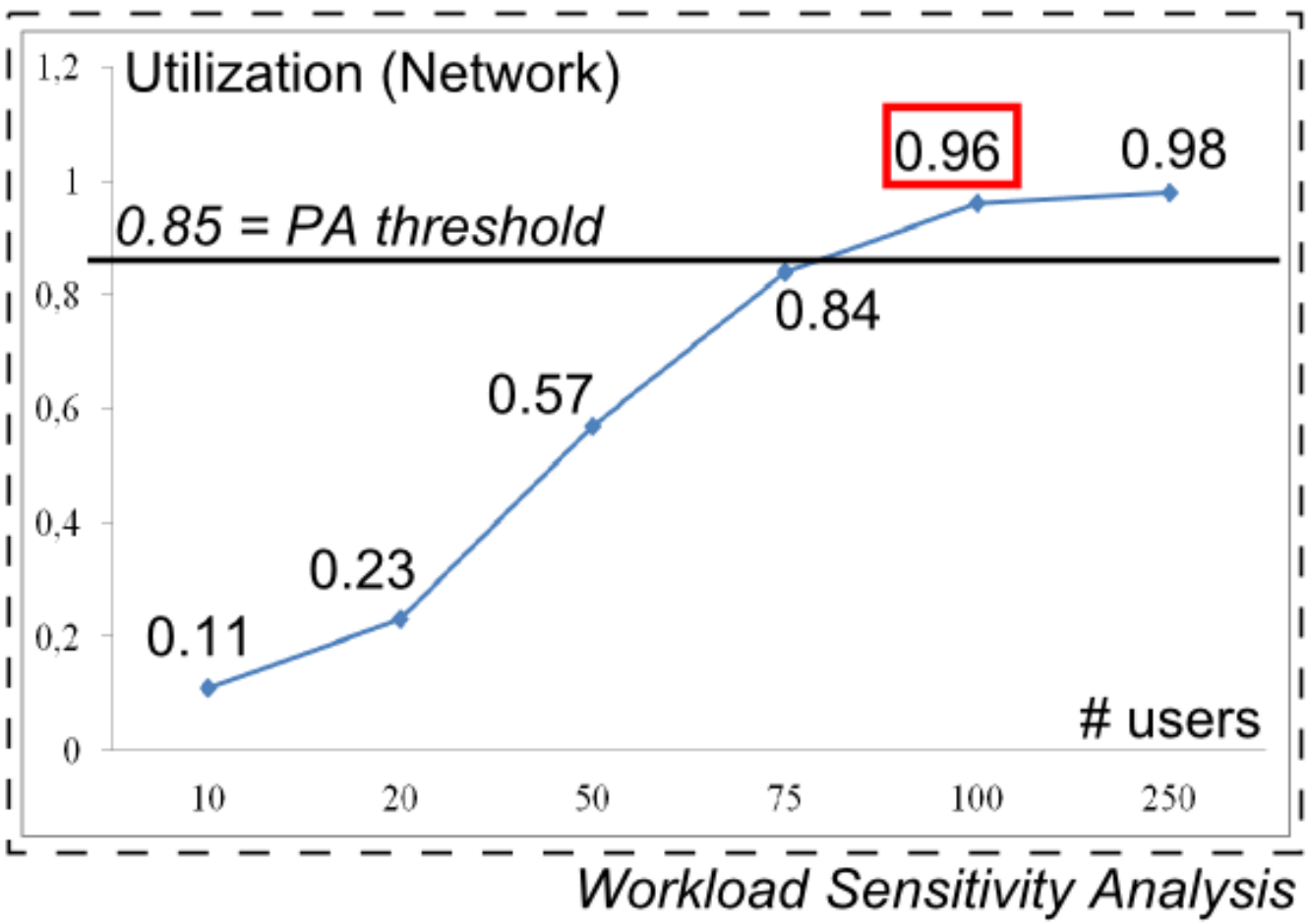
Detecting Antipatterns

antipattern	parameter	value
Blob	$Th_{maxConnect}$	4
	$Th_{maxMsgs}$	18
	$Th_{maxHwUtil}$	0.75
	$Th_{maxNetUtil}$	0.85
CPS	$Th_{maxQueue}$	40
	$Th_{cpuMaxUtil}$	0.8
	$Th_{diskMaxUtil}$	0.7
	$Th_{cpuMinUtil}$	0.3
	$Th_{diskMinUtil}$	0.4
EST	$Th_{remMsgs}$	12
	$Th_{remInst}$	5
	$Th_{minNetUtil}$	0.3
...	...	...

design properties
(e.g. *excessive* message traffic)

performance results
(e.g. *high* network utilization)

Workload: a trouble-making factor



» Performance Antipattern instances in ECS:

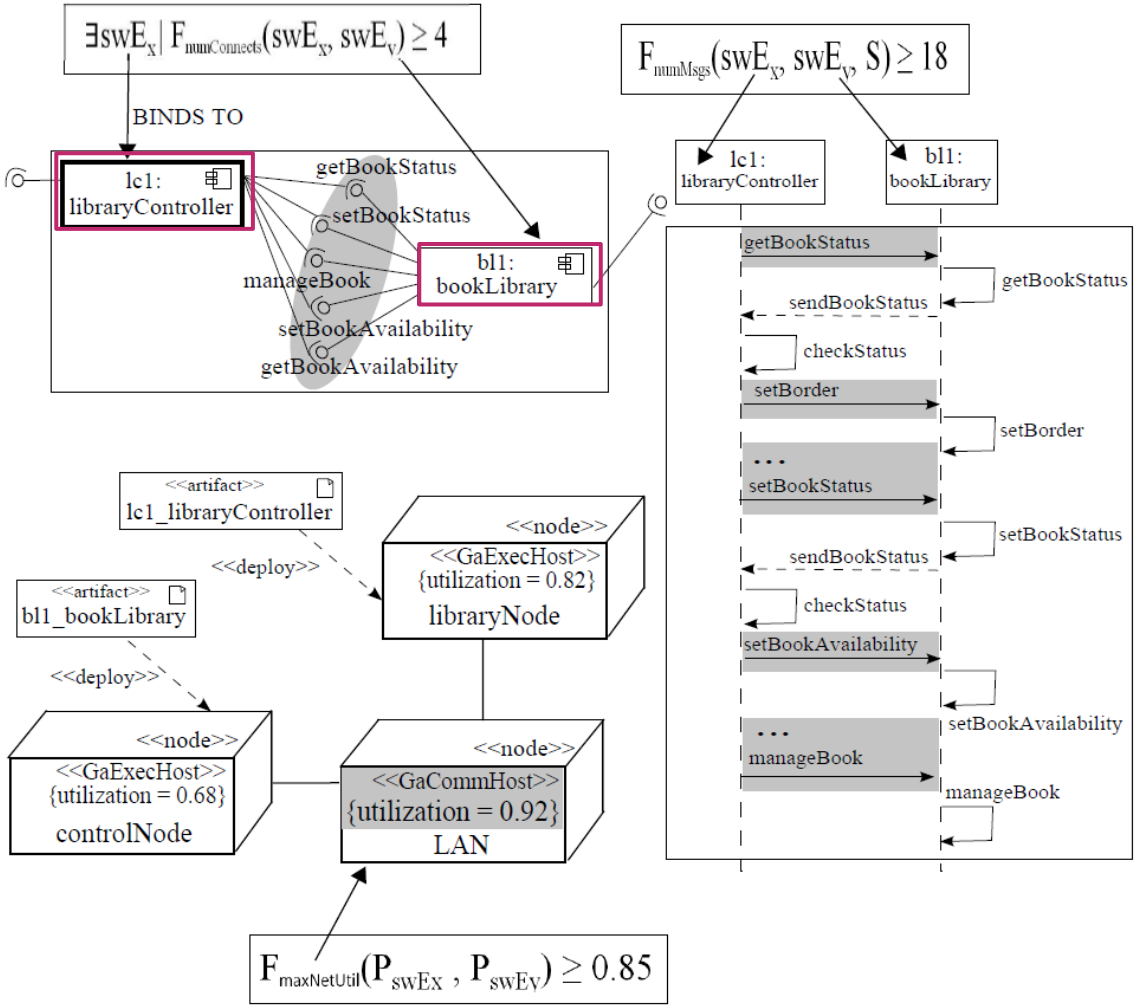
Detecting Antipatterns

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

» An example: detecting the Blob antipattern instance

Detecting Antipatterns

The (libraryController, bookLibrary, browseCatalog) instance satisfies the **Blob** predicate, hence it must be pointed out to the designer for a deeper analysis



And another Key-Question

Once we have (somehow) "detected" antipatterns,

which refactoring actions must be taken

to remove (some of) them?

First rough approach

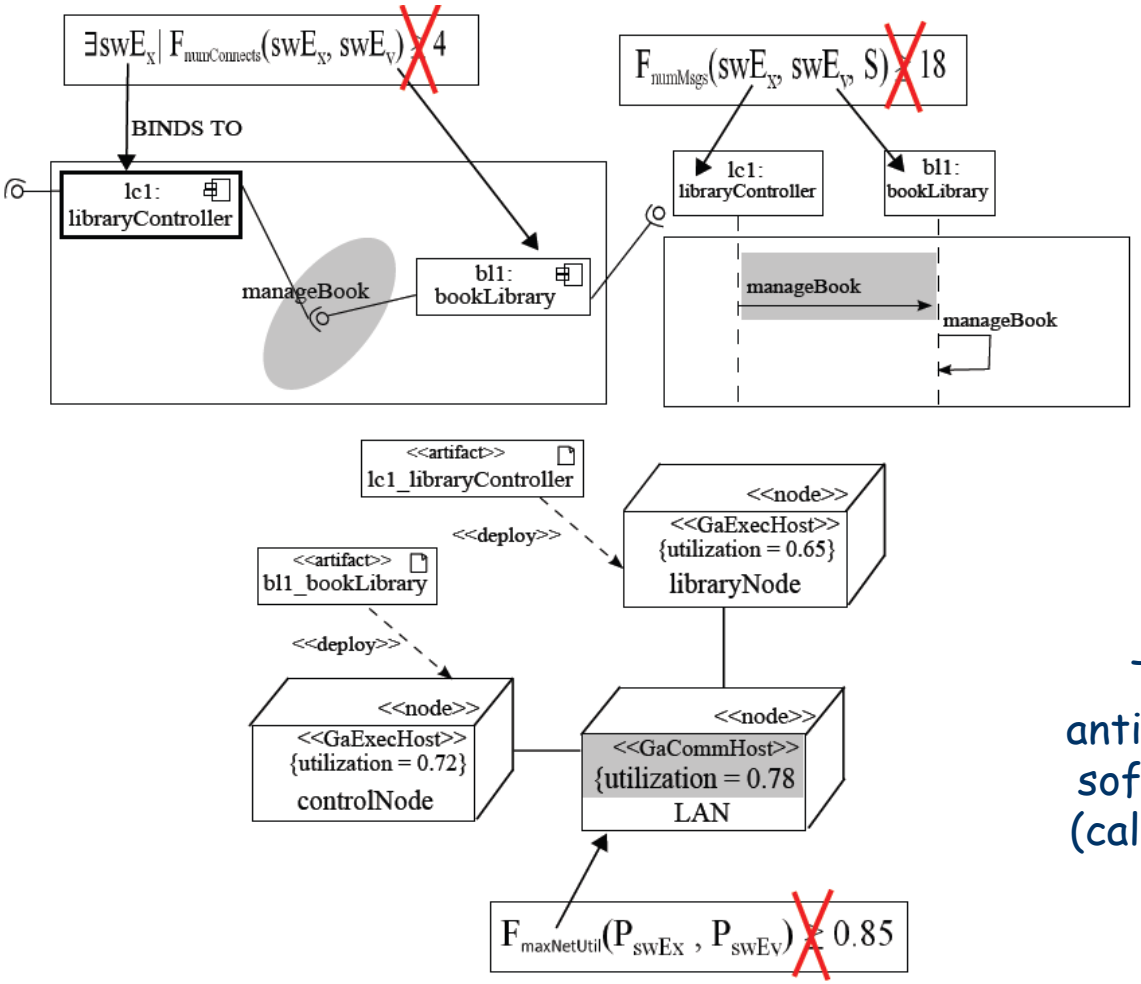
First-Order-Logic representation of antipatterns can help:

refactoring actions can be (automatically)

obtained from negating predicates!

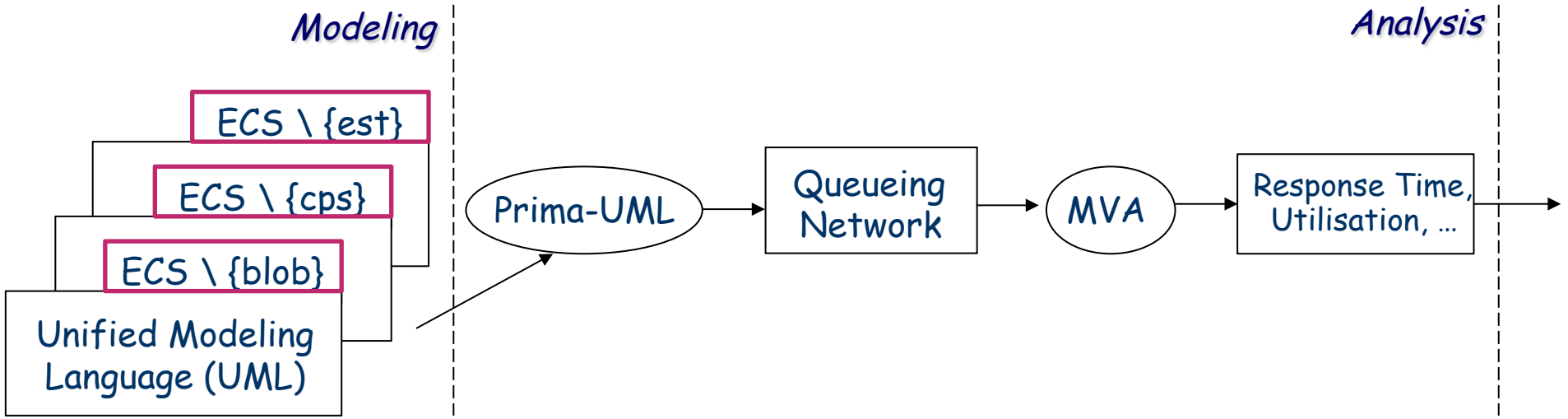
» An example: solving the Blob antipattern instance

Solving Antipatterns



The removal of the Blob antipattern gives rise to a new software architectural model (called here as "ECS \ {blob}")

Performance Analysis of **some** refactored models



(Annotated) Software Architectural Model	RT(browseCatalog) < = 1.2 sec	RT(makePurchase) < = 2 sec
ECS	1.5 sec	2.77 sec
ECS \ {blob}	1.14 sec	2.18 sec
ECS \ {cps}	1.15 sec	1.6 sec
ECS \ {est}	1.5 sec	2.24 sec

Yet another Key-Question??? Yes!

How to drive the process of antipattern solution?

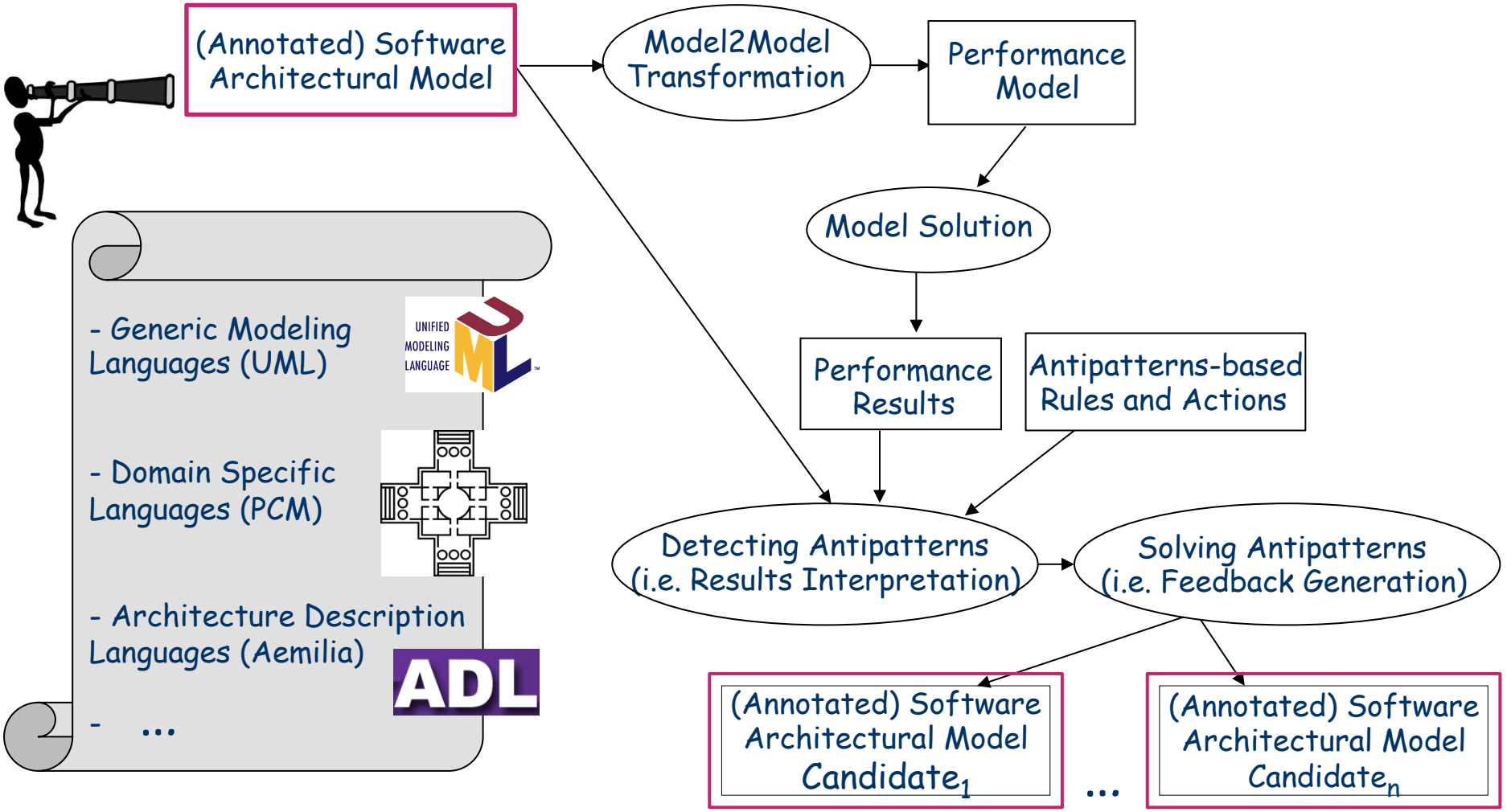
Who guarantees that the removal of antipatterns will lead to "better" performance?

First approach

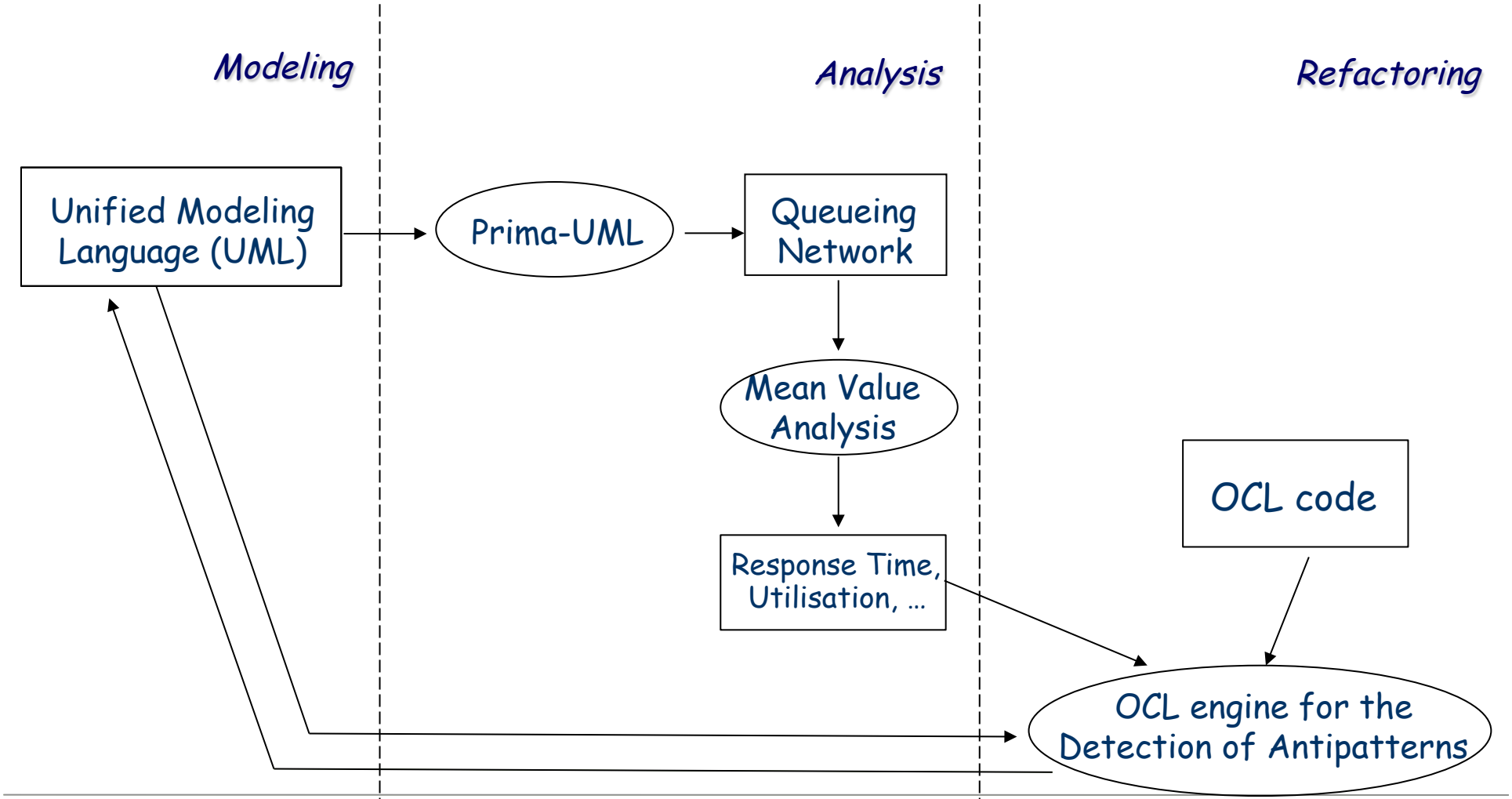
A guilt-based one, but still many issues to solve!

PERFORMANCE ANTIPATTERNS IN MODELING LANGUAGES

» Antipattern-based process



» Our experience in UML



» UML model

	Service Demand (input parameters)	Utilization (output indices)
webServerNode	2.02 msec	27%
libraryNode	7.05 msec	96%
controlNode	3 msec	41%
dbNode_cpu	15 msec	20%
dbNode_disk	30 msec	41%

Requirement: each hardware resource has not to be used more than 80% under the mean workload of 70 requests/second concurrently in execution in the system.

» An example: the Blob antipattern as OCL rule

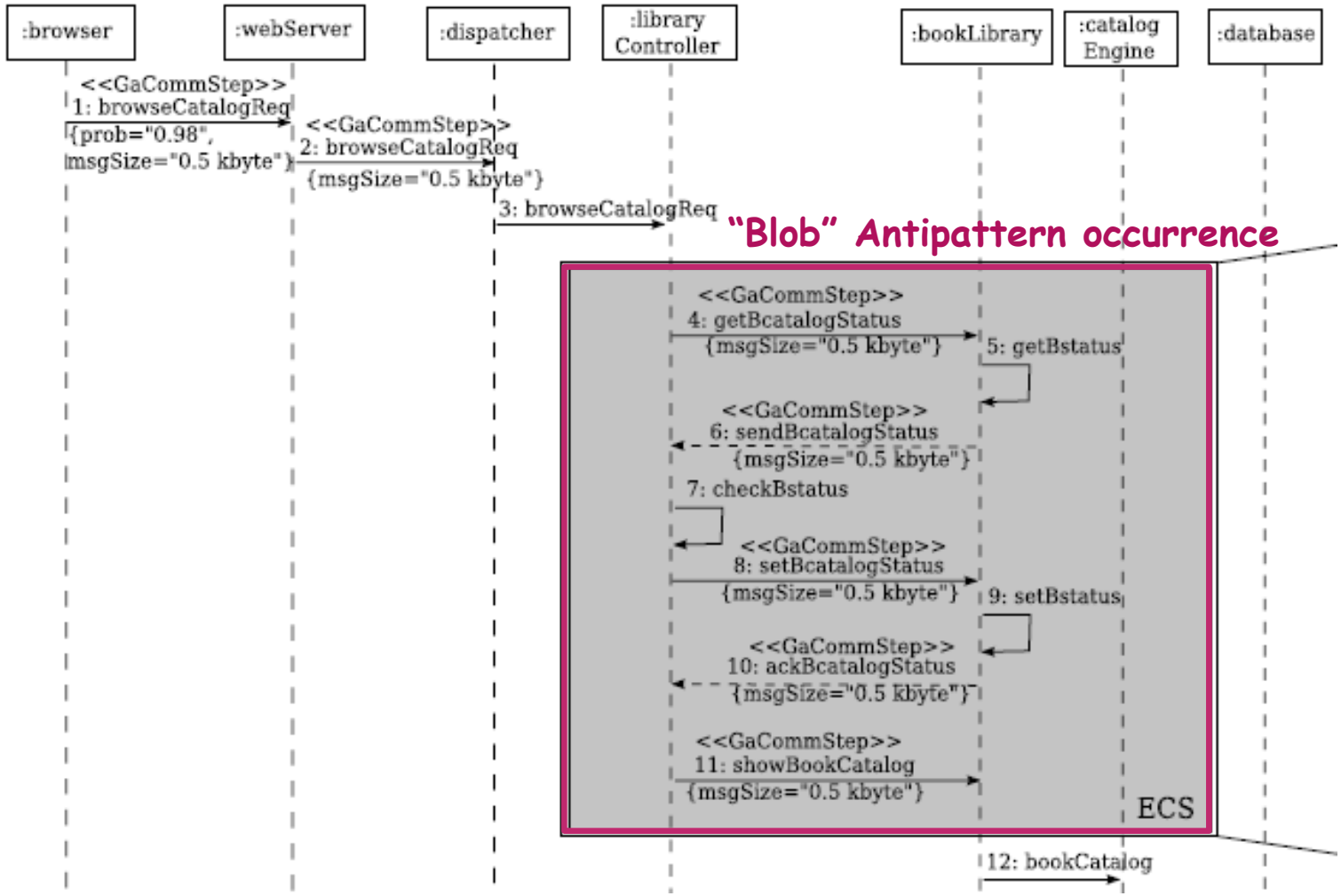
```
context Model ::
Blob() : Set(Component)

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

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

```
context Compon
usageRule() :

body : cc -> s
       >= getC
          getC
```

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

def: allLifelines: Set(L
     self.allOwnedElement
     ->select(oclIsTypeO
       .oclAlType(Lifeli

body : self ->select (ge
       -> select(getSentI
            > allLifeline
            / allLifeline
     )
```

```
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
     )
```

» Detecting antipatterns in UML



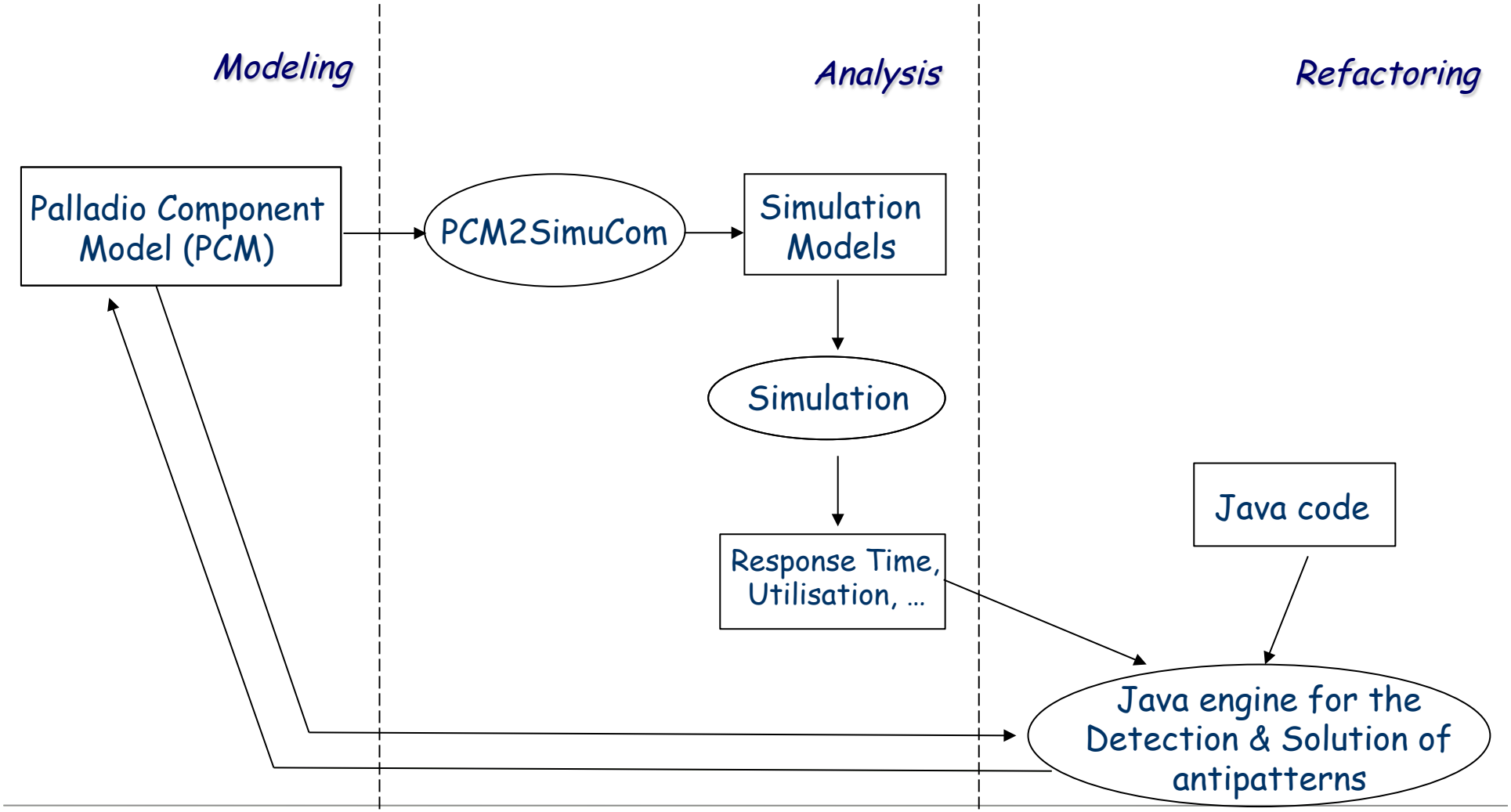
» Solving Antipatterns in UML

Notice that these values have changed due to the load re-distribution

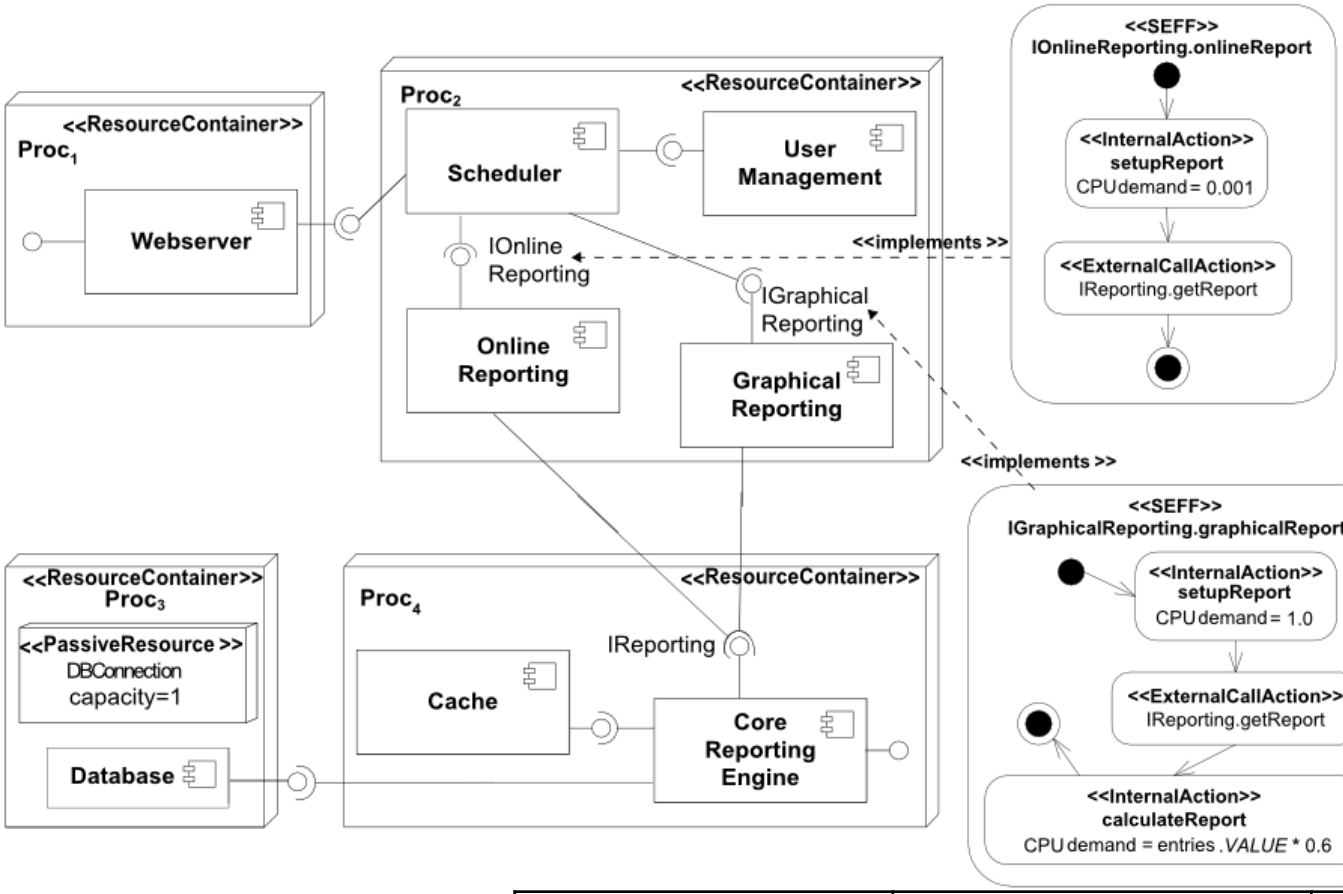
	Service Demand (input parameters)	Utilization (output indices)
webServerNode	4.07 msec	61%
libraryNode	5 msec	75%
controlNode	3 msec	41%
dbNode_cpu	15 msec	20%
dbNode_disk	30 msec	41%

Requirement: each hardware resource has not to be used more than 80% under the mean workload of 70 requests/second concurrently in execution in the system.

» Our experience in PCM



» PCM model



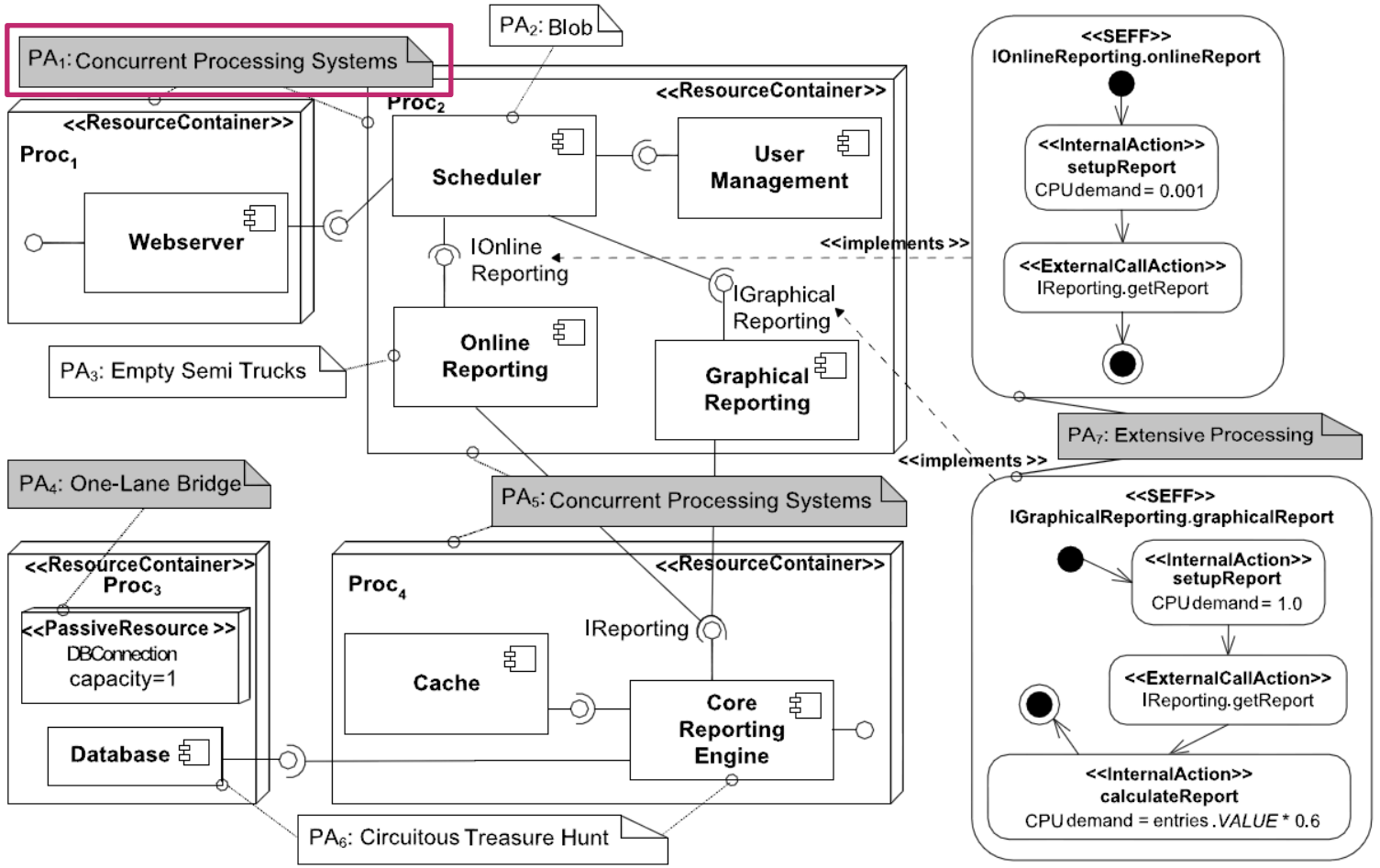
Requirement	Required Value	Predicted Value
RT(system)	10 sec	18.71 sec

» An example: the Concurrent Processing Systems antipattern as Java code

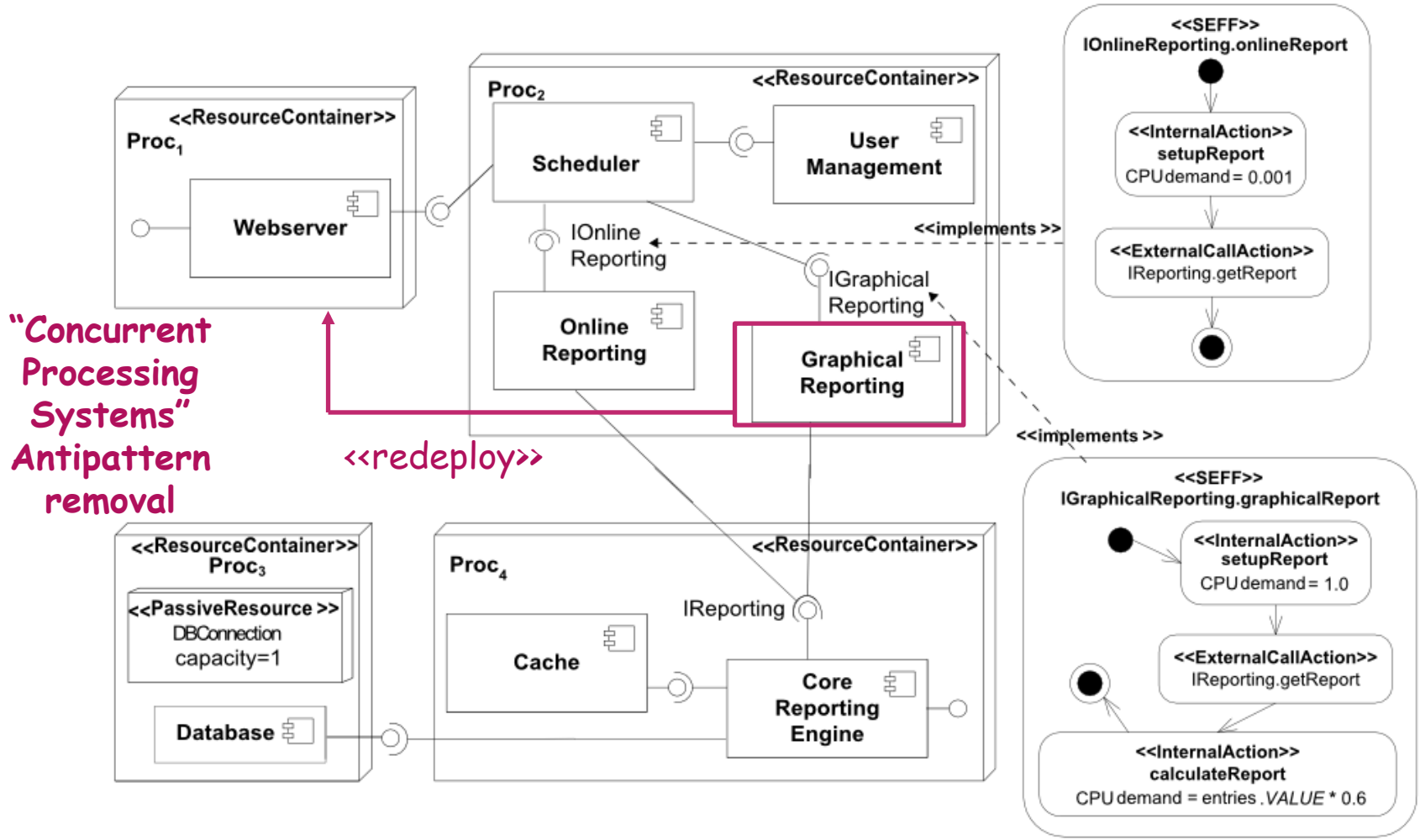
```
public boolean cps(List<ActiveResInfo> list) {
    boolean result = false;
    if ((getOverUsedCpu(list).size() != 0 && getUnderUsedCpu(list).size() != 0)
            || (getOverUsedHDD(list).size() != 0 && getUnderUsedHDD(list).size() != 0)) {
        logger.info("The antipattern CONCURRENT PROCESSING SYSTEMS has been detected");
        result = true;
    }
    return result;
}

public List<ActiveResInfo> getOverUsedCpu(List<ActiveResInfo> list){
    List<ActiveResInfo> result = new ArrayList<ActiveResInfo>(0);

    //@author catia: threshold value for
    double thresholdMaxCpu = new Thresho

    // @author catia: threshold value fo
    double thresholdCpuQL = new Threshol

    for (ActiveResInfo el : list) {
        if((el.type.getEntityName().equa
                (el.utilisation > thresh
                    (el.queueLength > th
            result.add(el);
        }
    }
    return result;
}

public List<ActiveResInfo> getUnderUsedCpu(List<ActiveResInfo> list){
    List<ActiveResInfo> result = new ArrayList<ActiveResInfo>(0);

    //@author catia: threshold value for min CPU utilisation
    double thresholdMinCpu = new Thresholds().thresholdMinCpu;

    for (ActiveResInfo el : list) {
        if((el.type.getEntityName().equals("CPU")) &&
                (el.utilisation < thresholdMinCpu)){
            result.add(el);
        }
    }
    return result;
}
```

» Detecting antipatterns in PCM

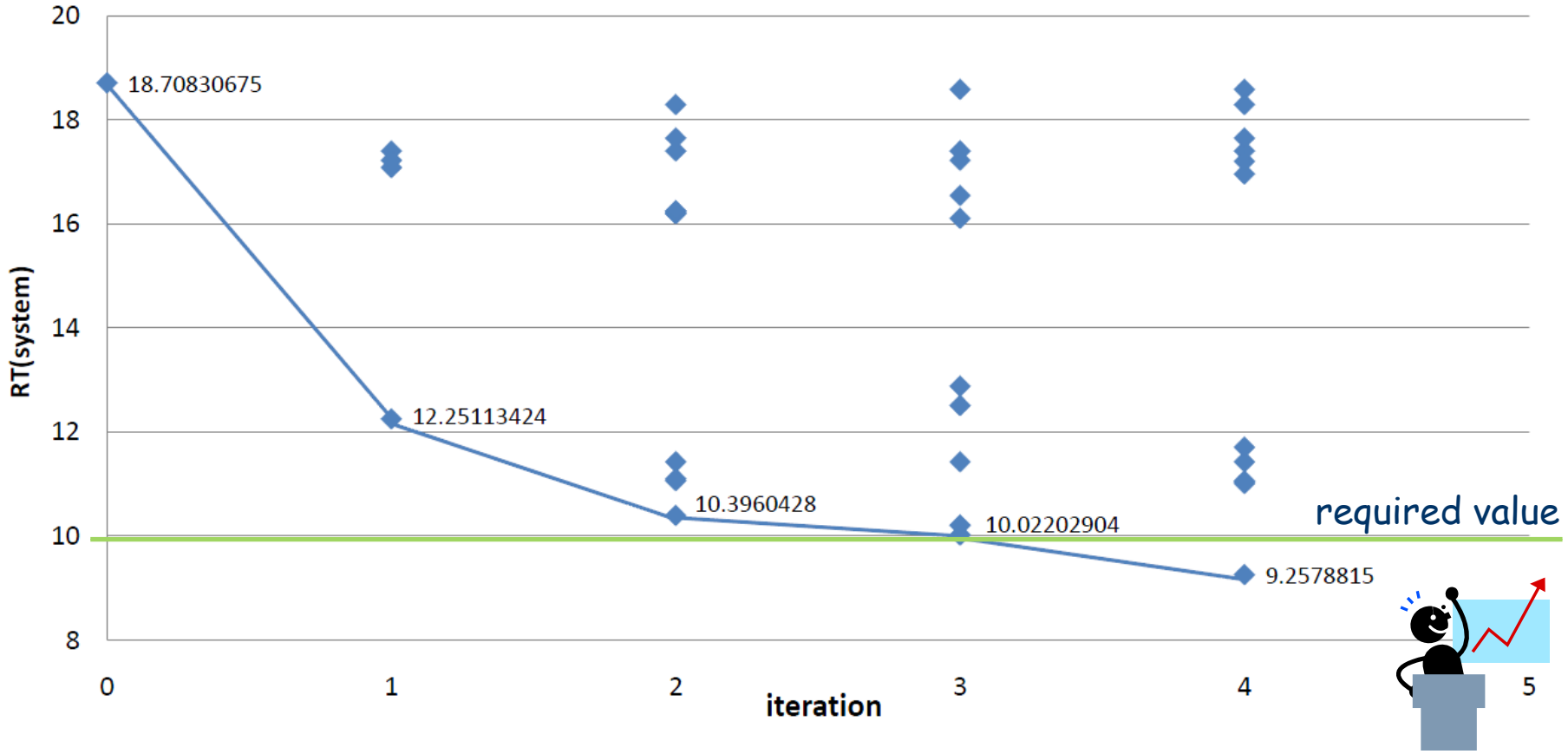
"Concurrent Processing Systems" Antipattern occurrence



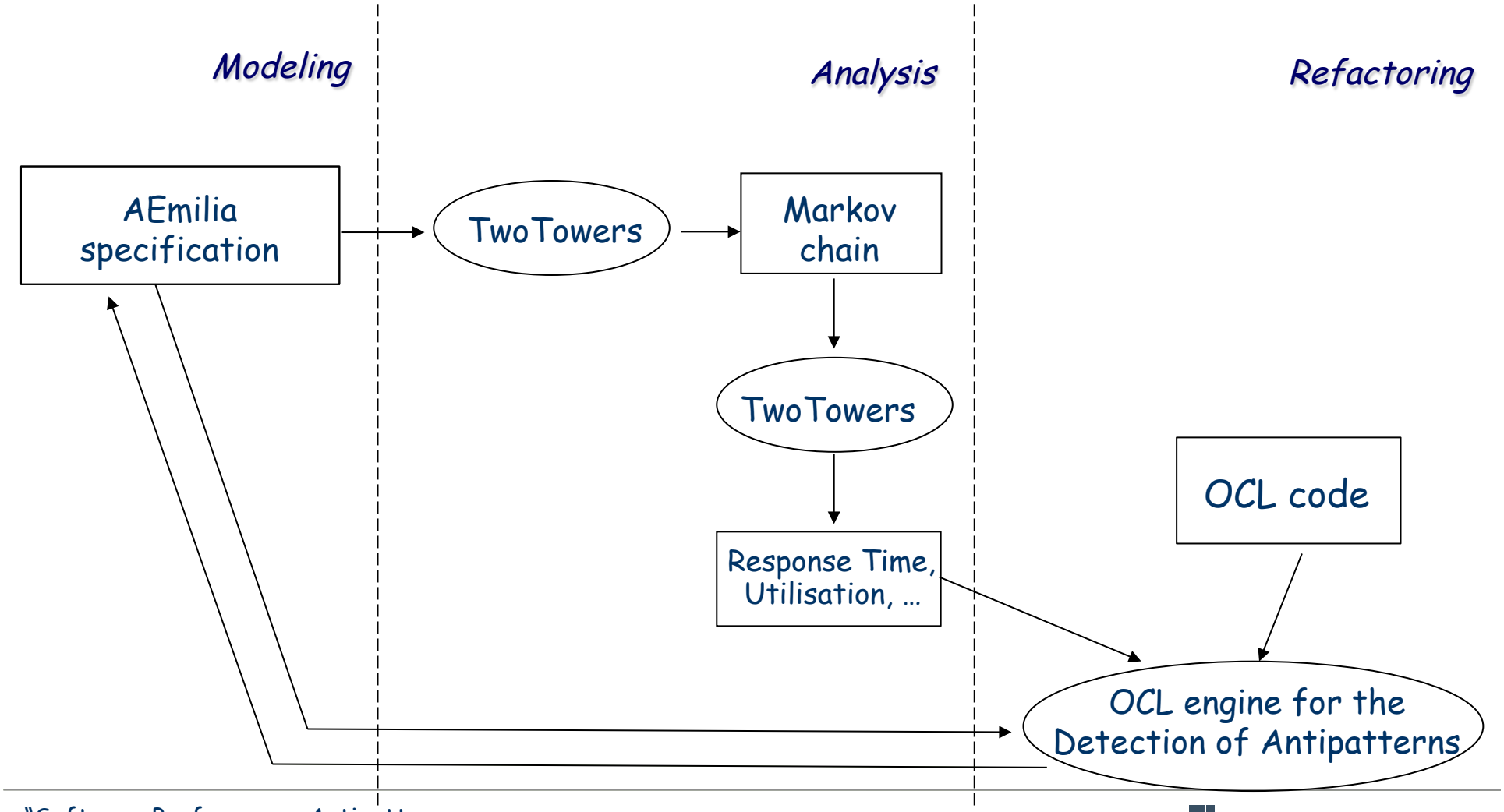
» Solving antipatterns in PCM



» The solution process in PCM



» Our experience in AEmilia



» AEmilia specification

```
ARCHI_TYPE boa( const integer ma_num := 5,
    const rate download_rate :=2441.40625,
    const rate upload_rate :=305.17578125,
    const rate balancer_rate_a :=20000000,
    const rate balancer_rate_b :=10000000,
    const rate server_req_rate:= 70000000,
    const rate server_result_rate:= 85995,
    const rate data_fetch_rate:= 36.585,
    const integer buffer_size :=10)

ARCHI_ELEM_TYPES

  ELEM_TYPE MA_Type(void)

      BEHAVIOR

      MobileApp(void; void) =
        <generate_best_path_req, inf> . <trasmit_req_best_path, inf> .
          <receive_best_path, _> . MobileApp()

      INPUT_INTERACTIONS

      UNI receive_best_path;
      generate_best_path_req

      OUTPUT_INTERACTIONS

      UNI trasmit_req_best_path
```

```
ARCHI_TYPE <name and formal parameters>

ARCHI_ELEM_TYPE
    <architectural element types:
        behaviors and Interactions>

ARCHI_TOPOLOGY
    ARCHI_ELEM_INSTANCES <architectural element instances>
    ARCHI_INTERACTIONS <architectural interactions>
    ARCHI_ATTACHMENTS <architectural attachments>
```

Requirement	Required Value	Predicted Value
Throughput(system)	15 reqs/sec	12.19 reqs/sec

» An example: the Extensive Processing antipattern as OCL code (**applied on the Aemilia MM!**)

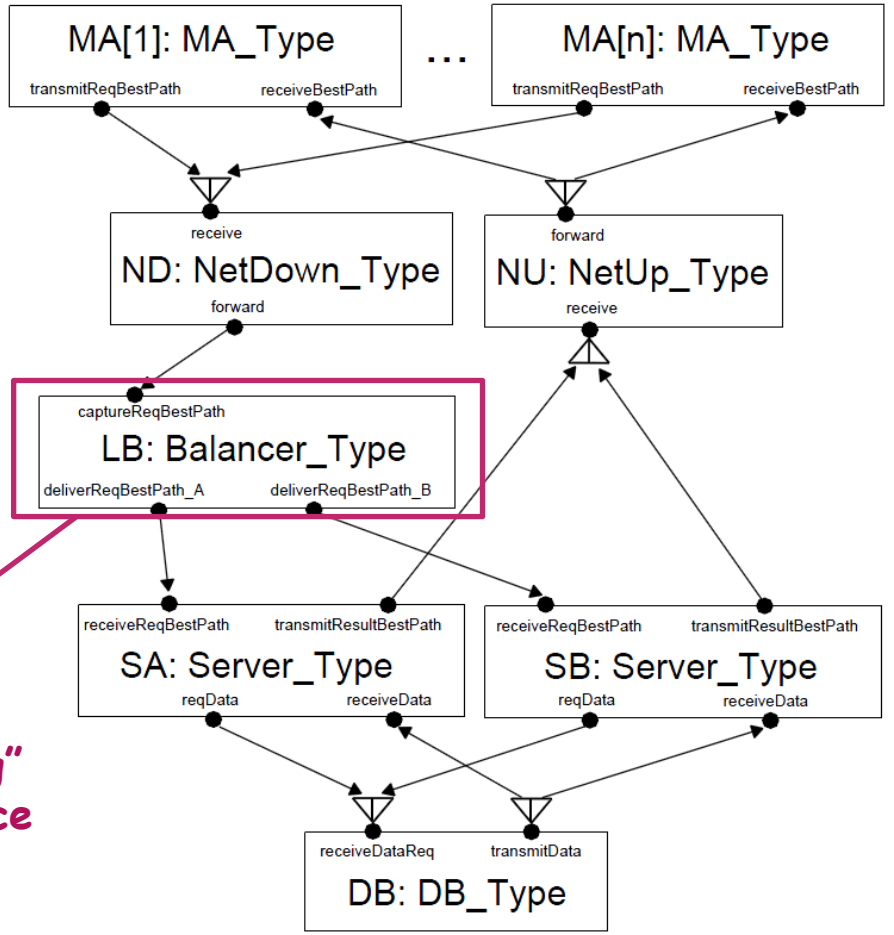
```
--function for the detection of the Extensive Processing antipattern
def: checkExtensiveProcessingCond(element: ElemType,
maxOpResDemand: Real, minOpResDemand: Real) : Boolean =
let opWithHighResDemand : Sequence(Behavior::Action) =
    findOpWithHighResDemand(element, maxOpResDemand) in
let opWithLowResDemand : Sequence(Behavior::Action) =
    findOpWithLowResDemand(element, minOpResDemand) in
if (opWithHighResDemand -> size() <> 0 and opWithLowResDemand ->size() <> 0) then
    opWithHighResDemand -> exists(act1: Behavior::Action |
    opWithLowResDemand -> exists(act2: Behavior::Action
                                | belongToTheSameChoice(act1, act2)))

else
    false
endif
```

```
def: findOpWithHighResDemand (elemType: ElemType, bound: Real) : Sequence(Behavior::Action) =
    Behavior::Action.allInstances() ->
                    select(act: Behavior::Action |
                    act.belongs.etName = elemType.etName and
                    act.rate.oclIsTypeOf(Behavior::RateExp) and
                    getActionRate(act) >= bound) -> asSequence()
```

» Detecting antipatterns in AEmilia

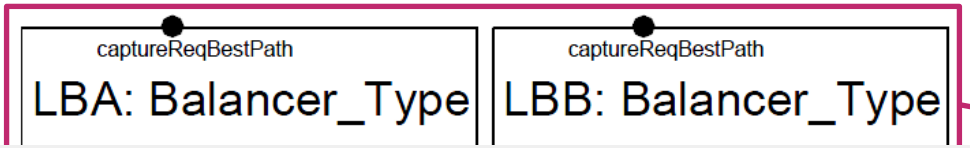
C:\Users\Catia\Desktop\BoA-caseStudy\BoA.aem

```
172
173 ARCHI_TOPOLOGY
174
175   ARCHI_ELEM_INSTANCES
176     FOR_ALL i IN 1..ma_num
177       MA[i] : MA_Type();
178     ND : NetDown_Type(buffer_size, downlo
179     NU : NetUp_Type(buffer_size, upload_r
180     LB : Balancer_Type(buffer_size, balan
181     SA : Server_Type(buffer_size, server_
182     SB : Server_Type(buffer_size, server_
183     DB : DB_Type(buffer_size, data_fetch_
```



"Extensive Processing" Antipattern occurrence

» Solving antipatterns in AEmilia

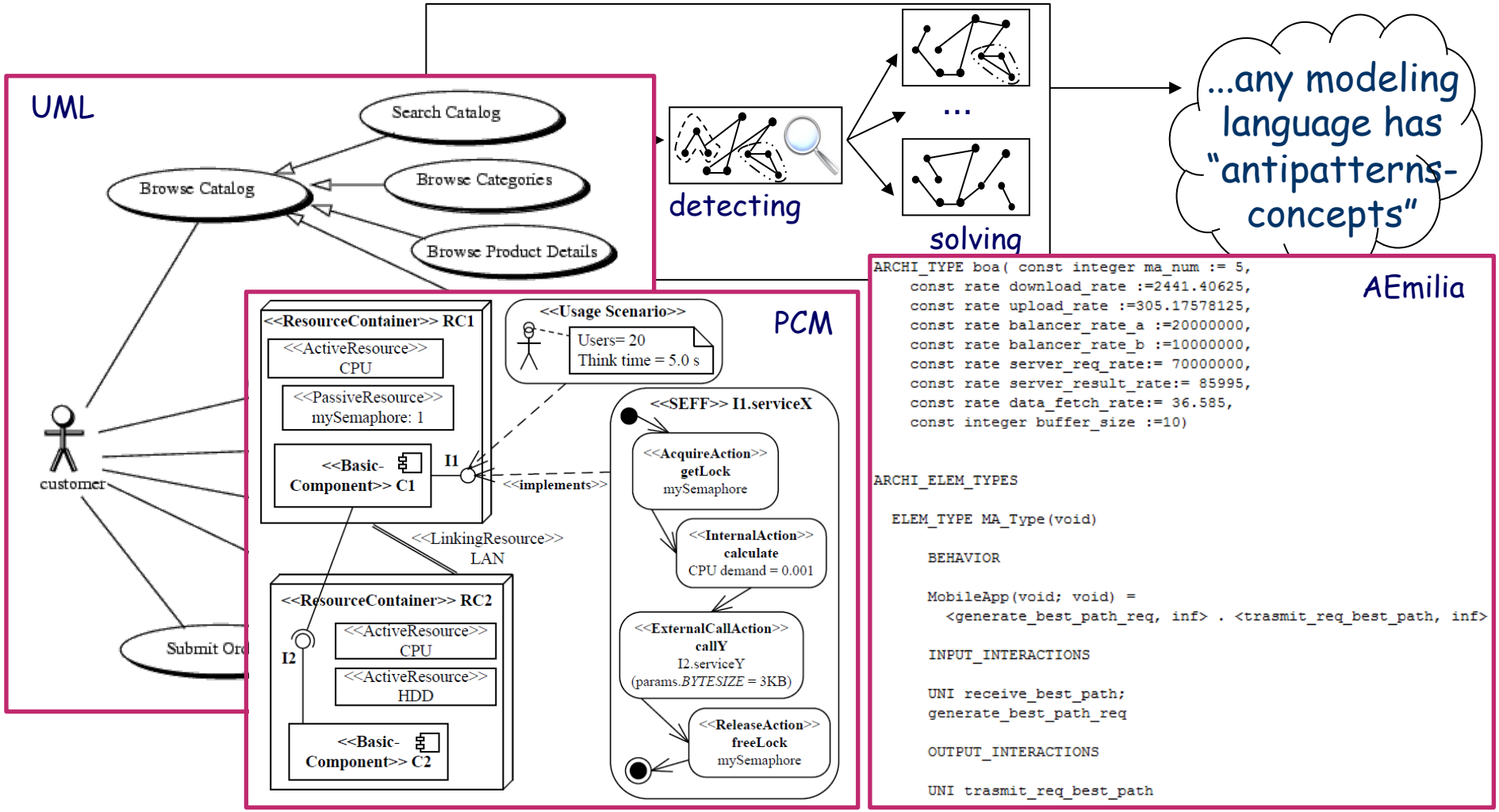


C:\Users\Catia\Desktop\BoA-caseStudy\BoA-Refactoring\BoA-EP.aem

```
170
171 ARCHI_TOPOLOGY
172
173   ARCHI_ELEM_INSTANCES
174     FOR_ALL i IN 1..ma_num
175       MA[i] : MA_Type();
176     ND : NetDown_Type(buffer_size, download_rate, routing_prob);
177     NU : NetUp_Type(buffer_size, upload_rate);
178     LBA : Balancer_Type(buffer_size, balancer_rate_a);
179     LBB : Balancer_Type(buffer_size, balancer_rate_b);
180     SA : Server_Type(buffer_size, server_req_rate, server_result_rate);
181     SB : Server_Type(buffer_size, server_req_rate, server_result_rate);
182     DB : DB_Type(buffer_size, data_fetch_rate)
```

Requirement	Required Value	Predicted Value
Throughput(system)	15 reqs/sec	18.29 reqs/sec

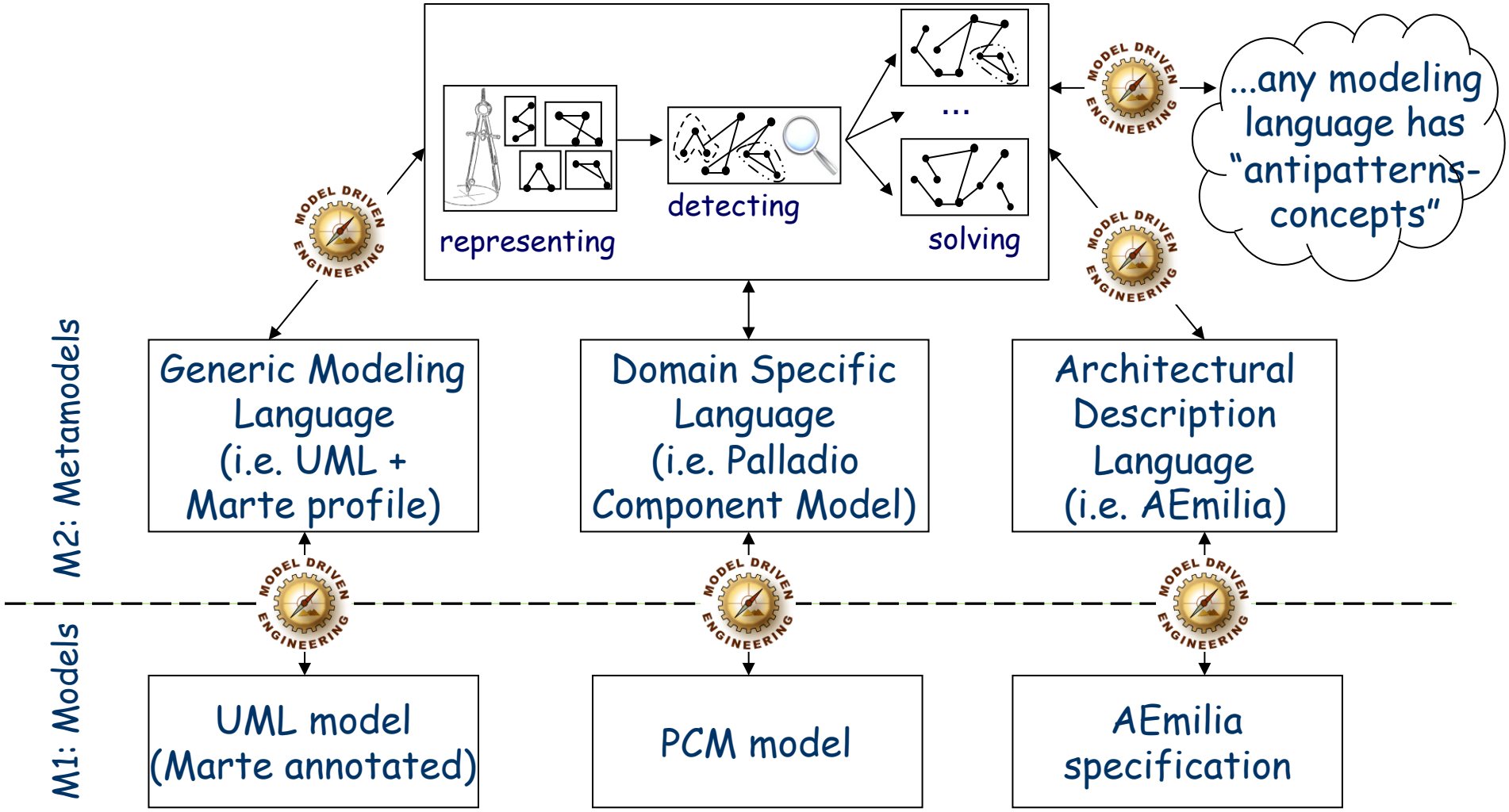
Looking at different modeling notations…



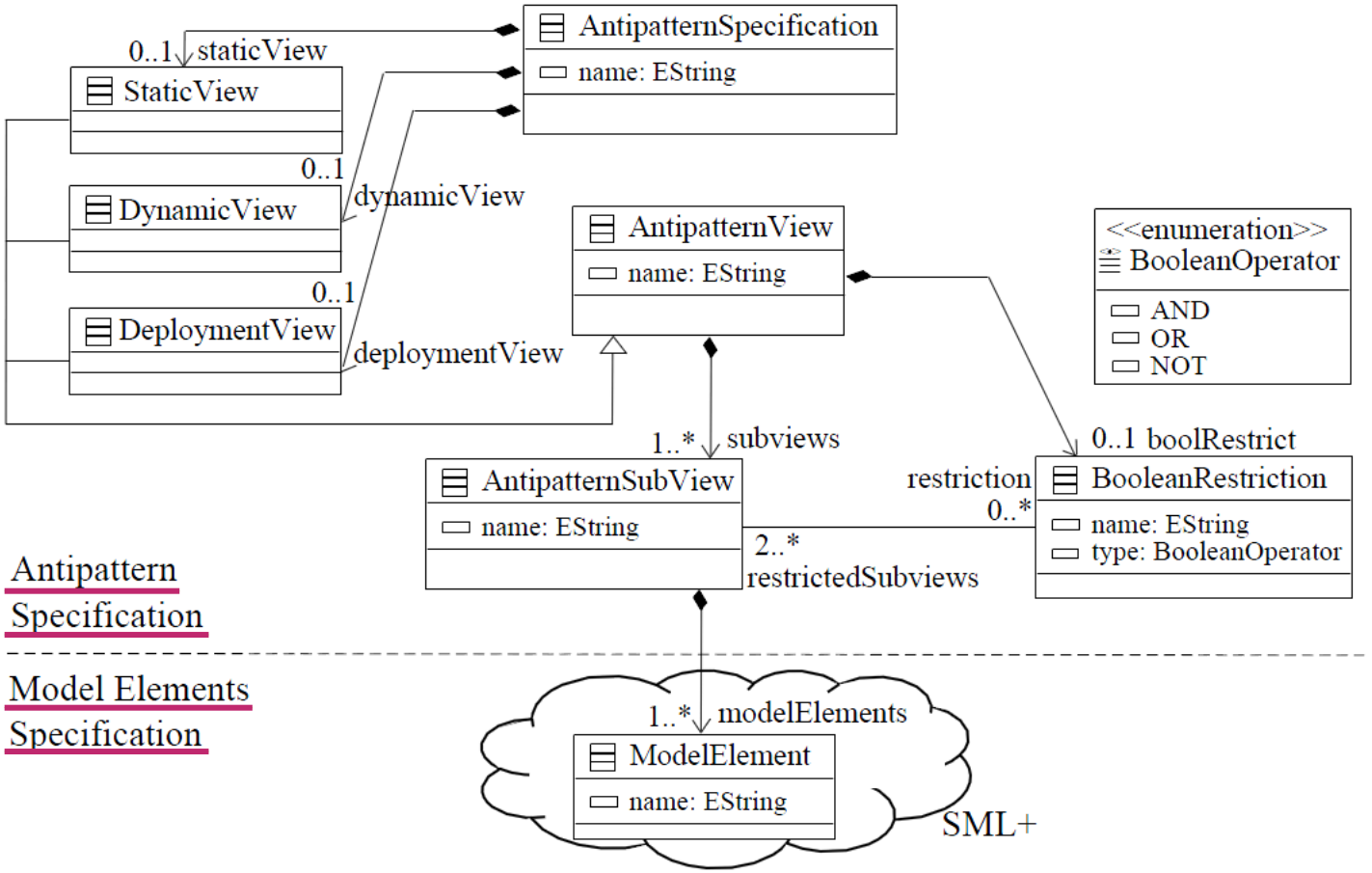
ADVANCED MODEL-DRIVEN TECHNIQUES…

TO TACKLE THE PROBLEM IN A LANGUAGE-INDEPENDENT WAY

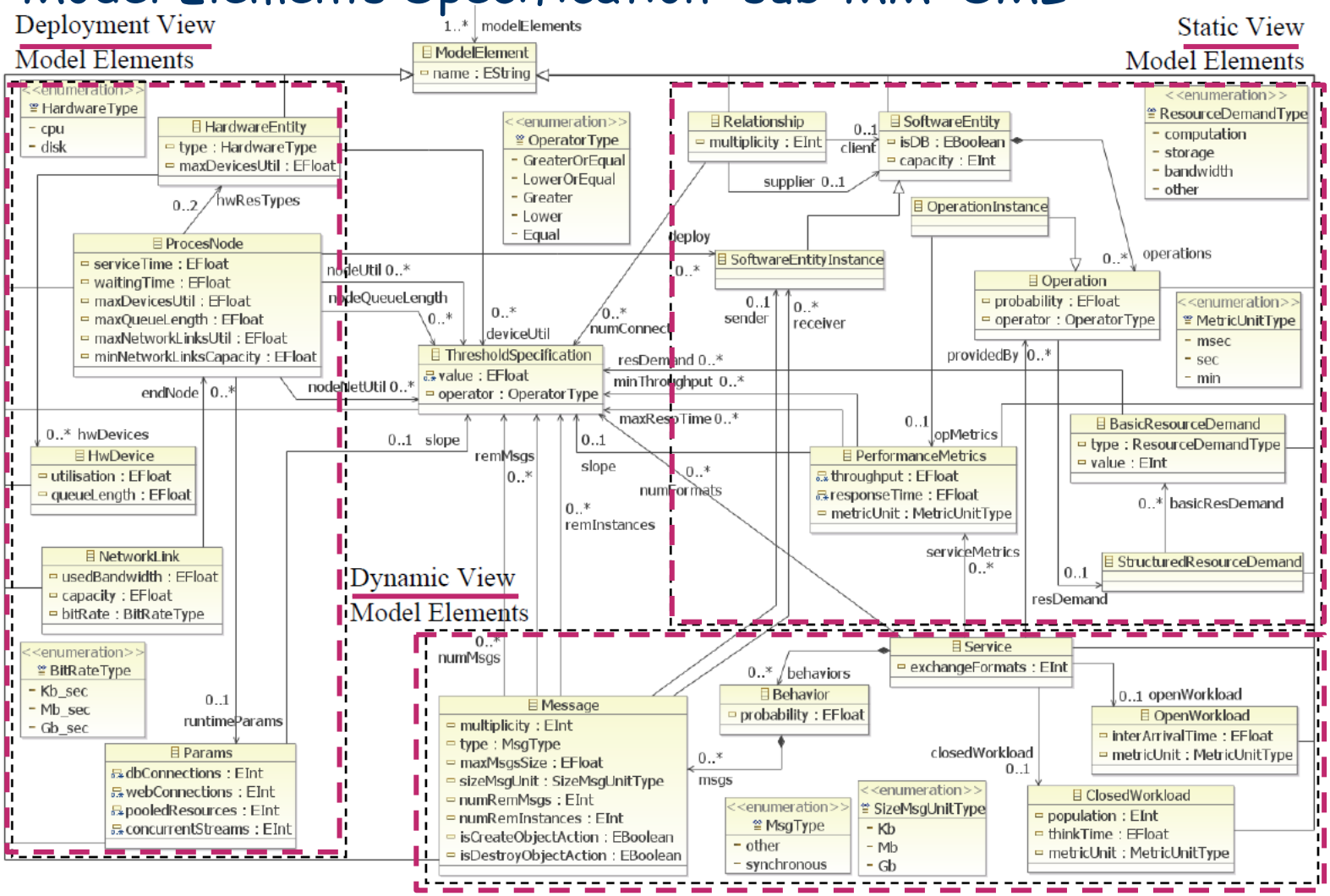
MDE support: metamodel and model transformations



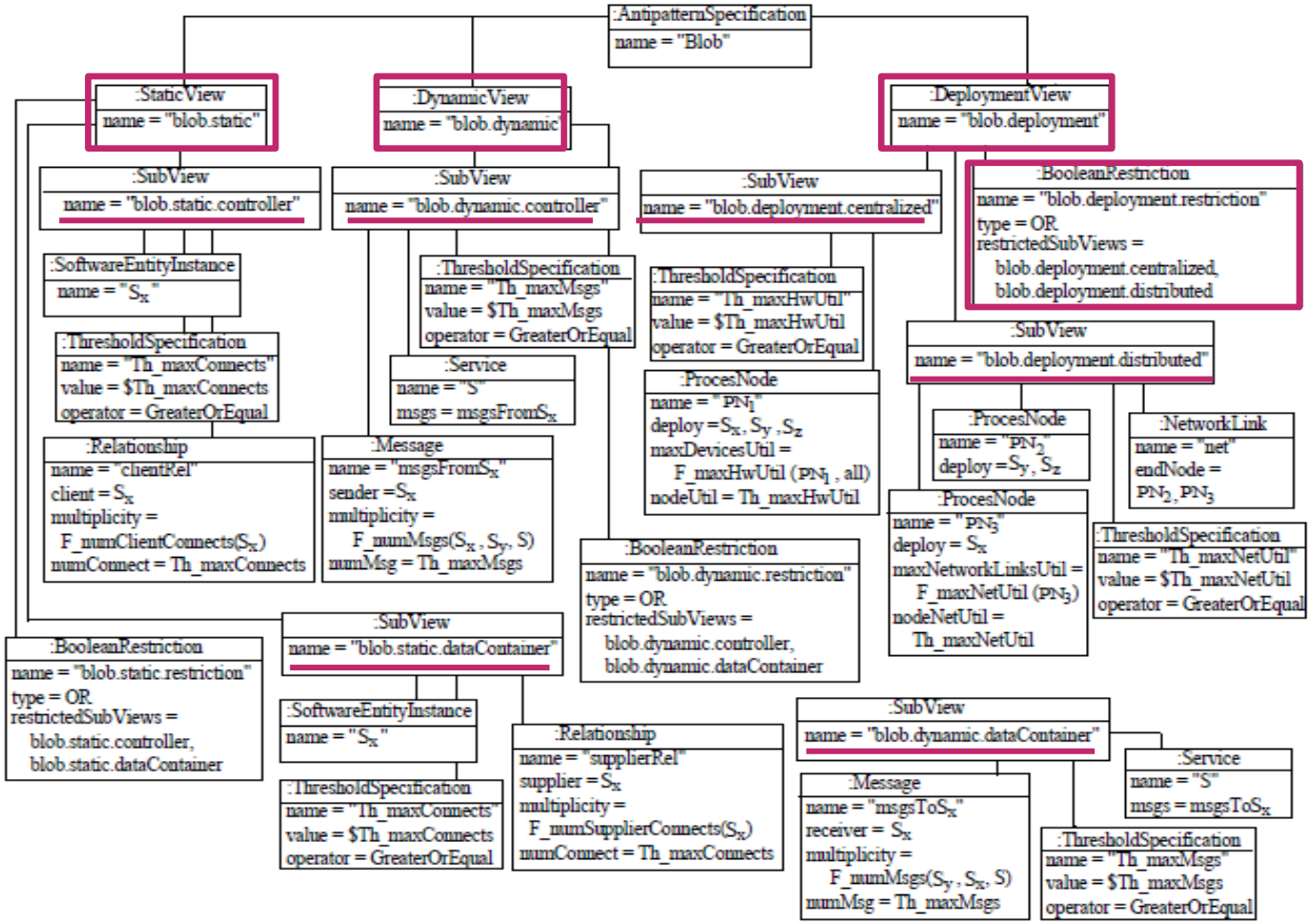
PAML: Perf.Antipatterns Modeling Language



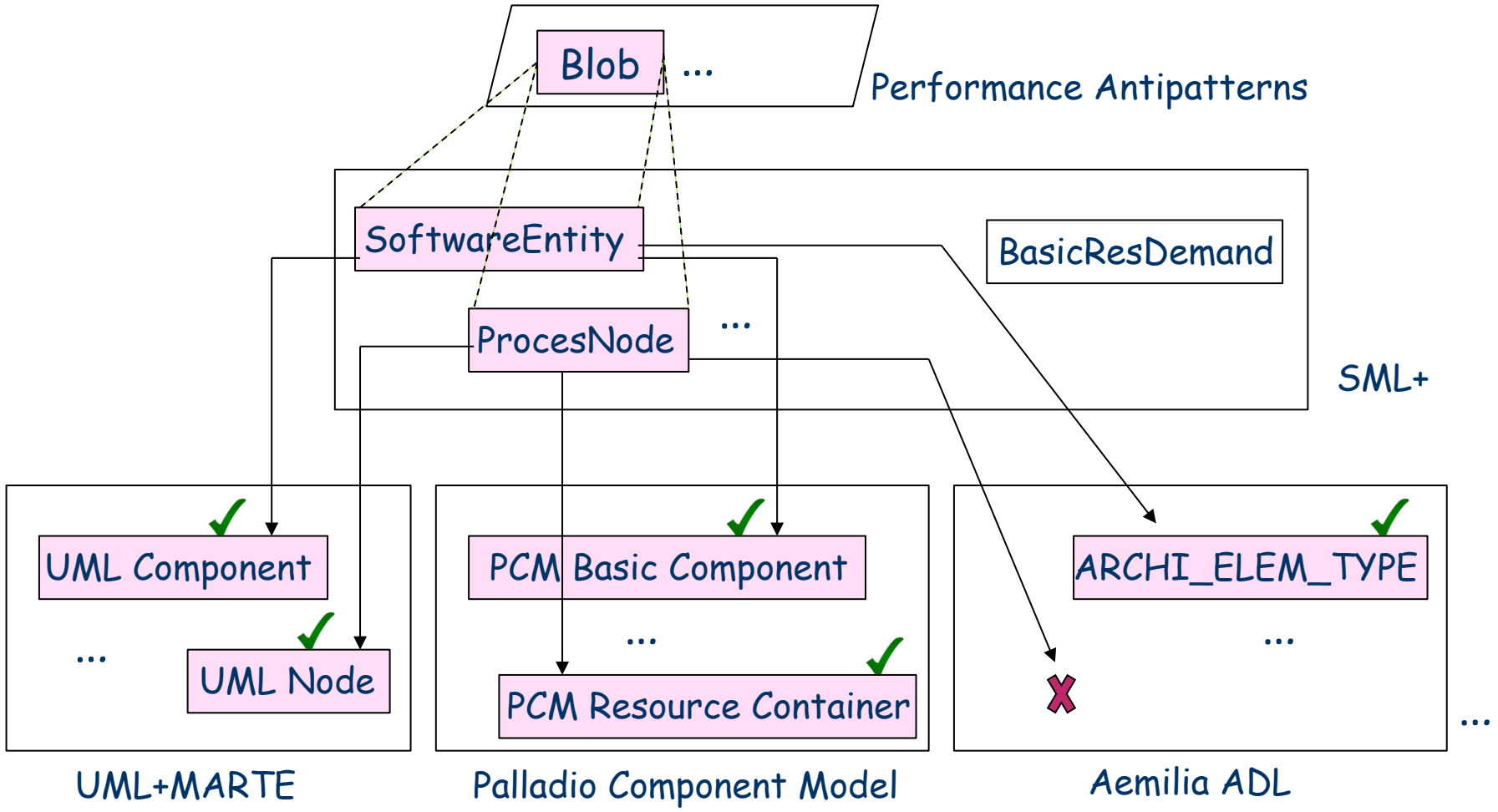
The "Model Elements Specification" sub-MM: SML+



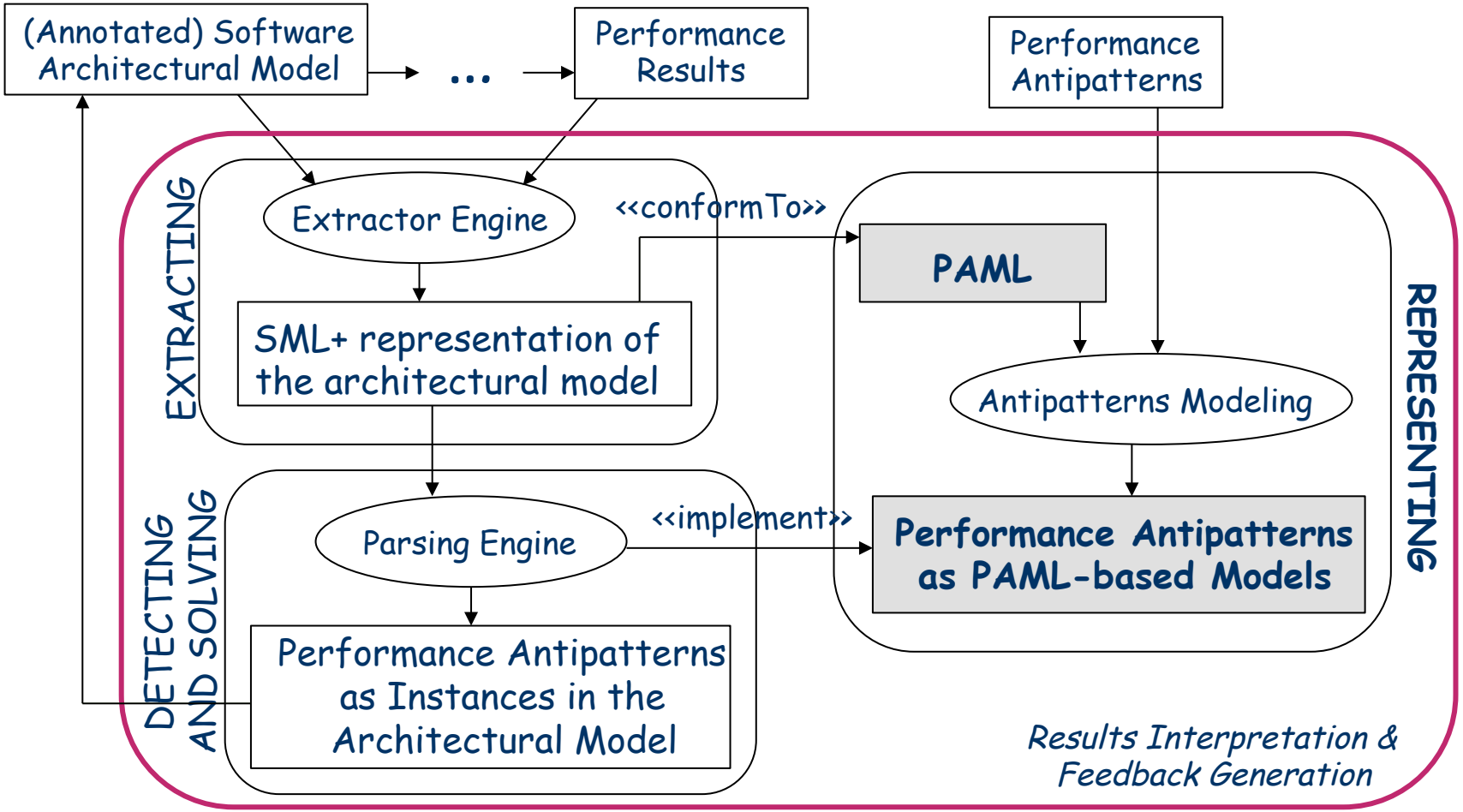
The "Blob" antipattern as PAML-based model



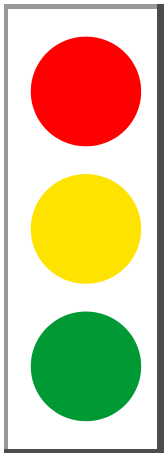
Antipatterns in concrete modeling languages

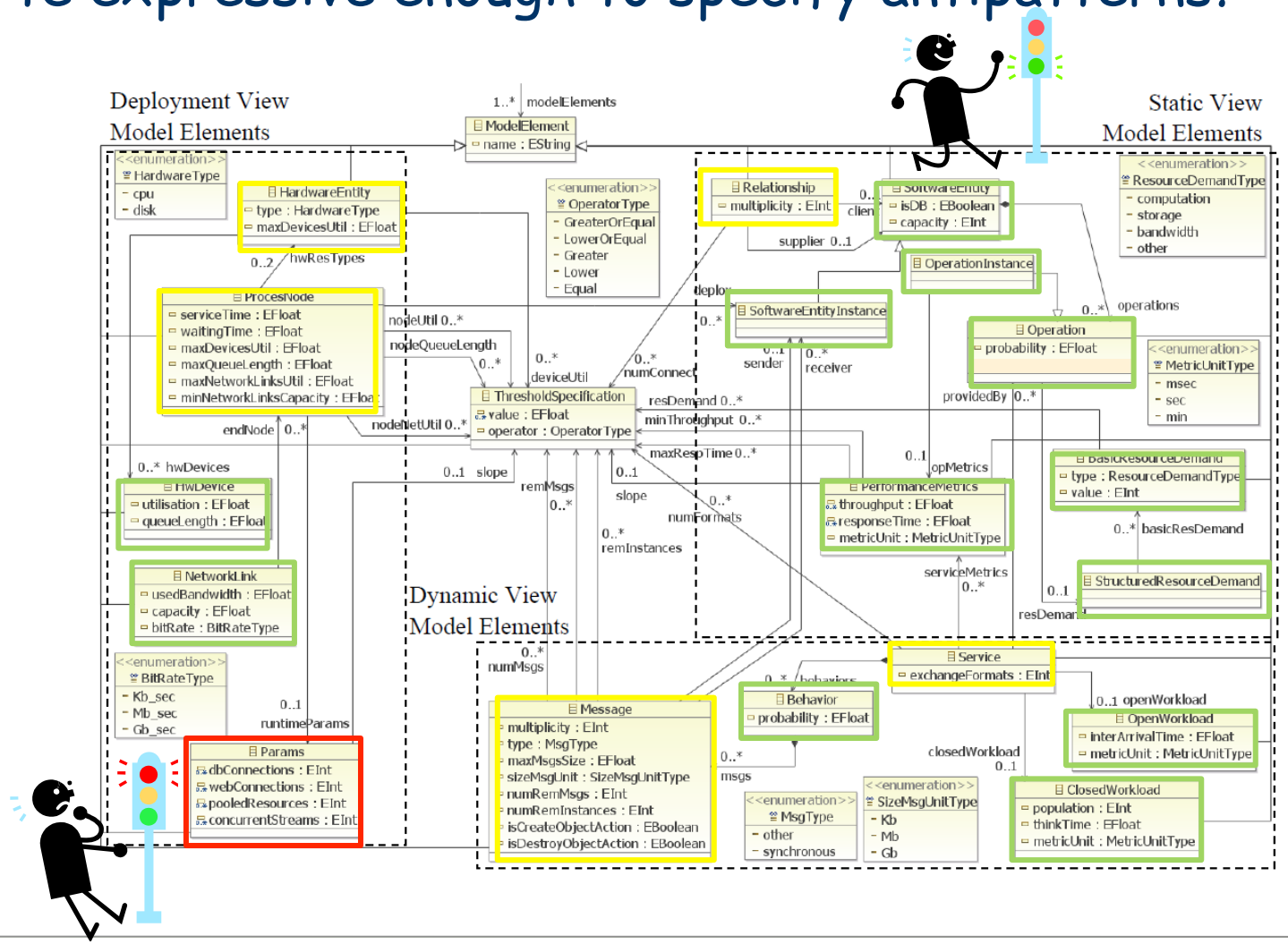


MDE approach for managing antipatterns

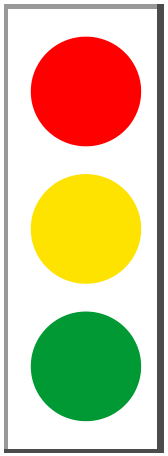


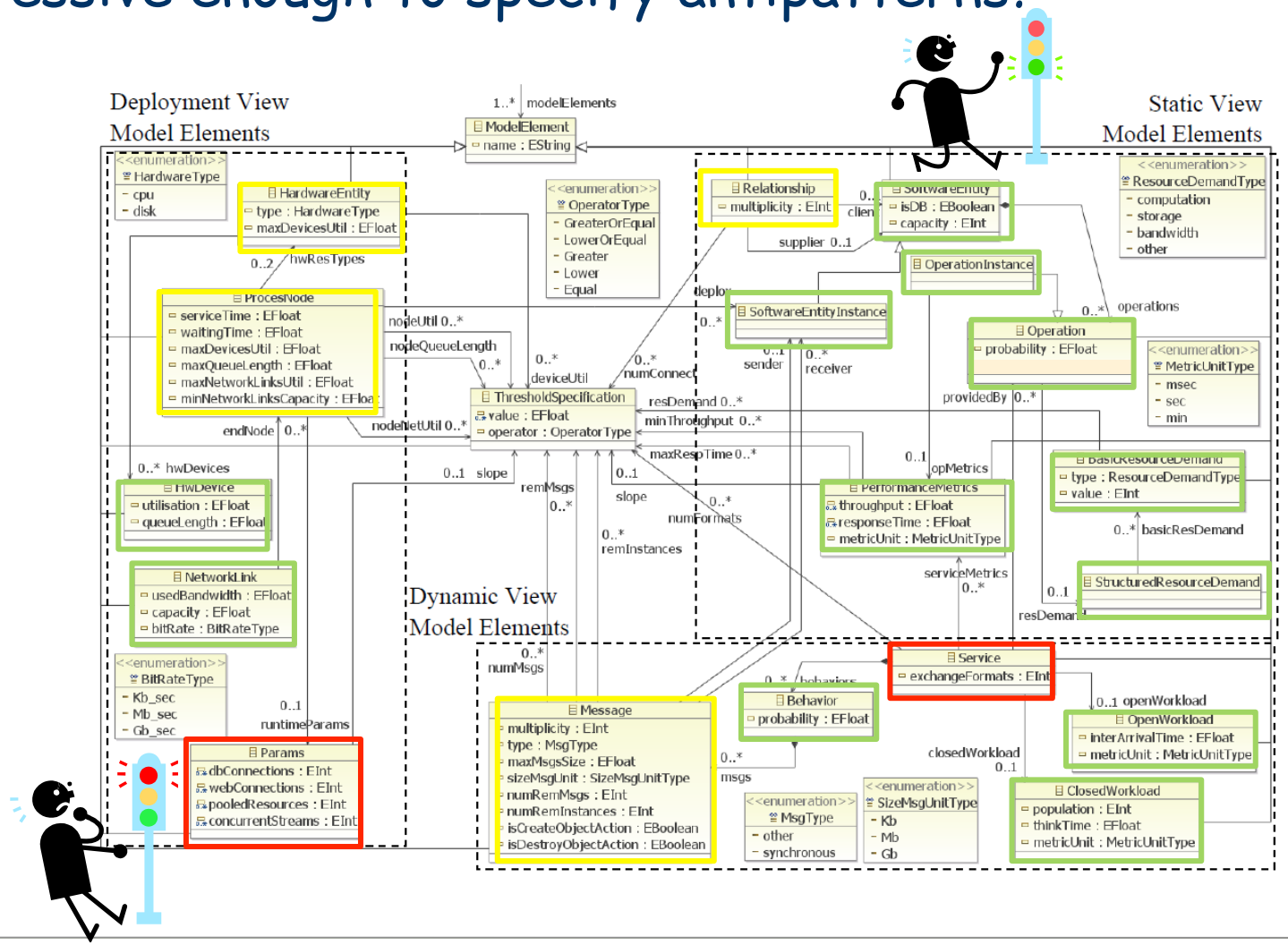
Is UML+Marte expressive enough to specify antipatterns?



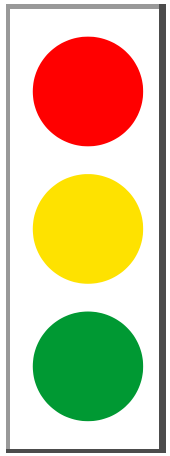


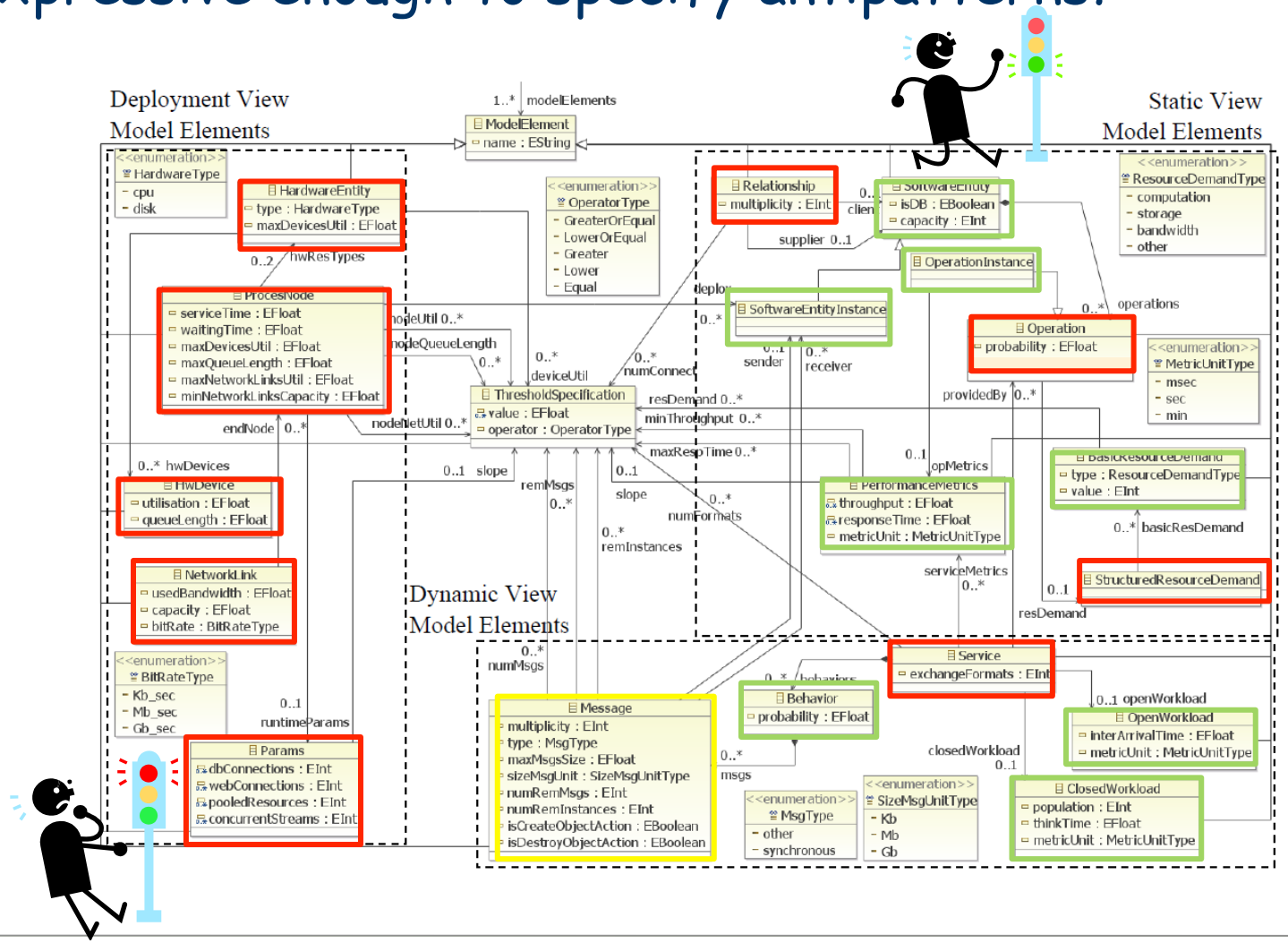
Is PCM expressive enough to specify antipatterns?





Is Aemilia expressive enough to specify antipatterns?





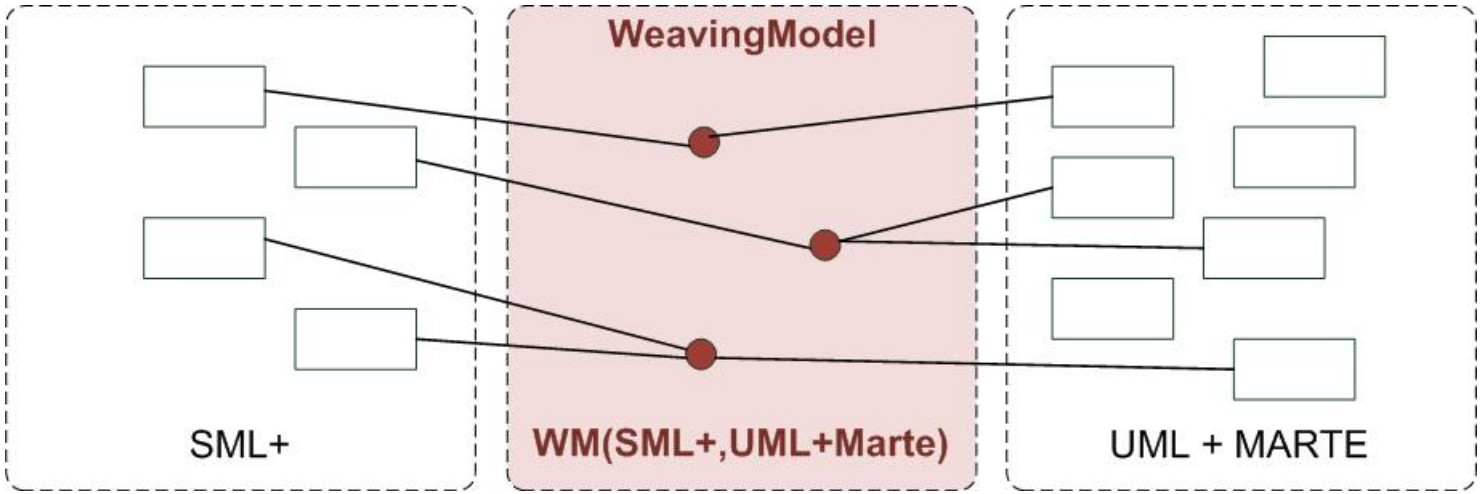
Expressiveness of considered modeling languages

		UML + Marte profile		Palladio Component Model		AEmilia	
Antipattern		Detectable	Solvable	Detectable	Solvable	Detectable	Solvable
Blob		✔	✔	✔	✔	≈	≈
Unbalanced Processing	Concurrent Processing Systems	✔	✔	✔	✔	✖	✖
	Pipe and Filter Architectures	✔	✔	✔	✖	✔	✔
	Extensive Processing	✔	✔	✔	✔	✔	✔
Tower of Babel		✔	✖	✖	✖	✖	✖
...		...	...	...	...	...	...
The Ramp		✔	✖	✔	✖	✔	✖
Traffic Jam		✔	✔	✔	✖	✔	✔
More is Less		✖	✖	✖	✖	✖	✖

✔ = fully detectable/solvable ≈ = partially detectable/solvable ✖ = not detectable/solvable

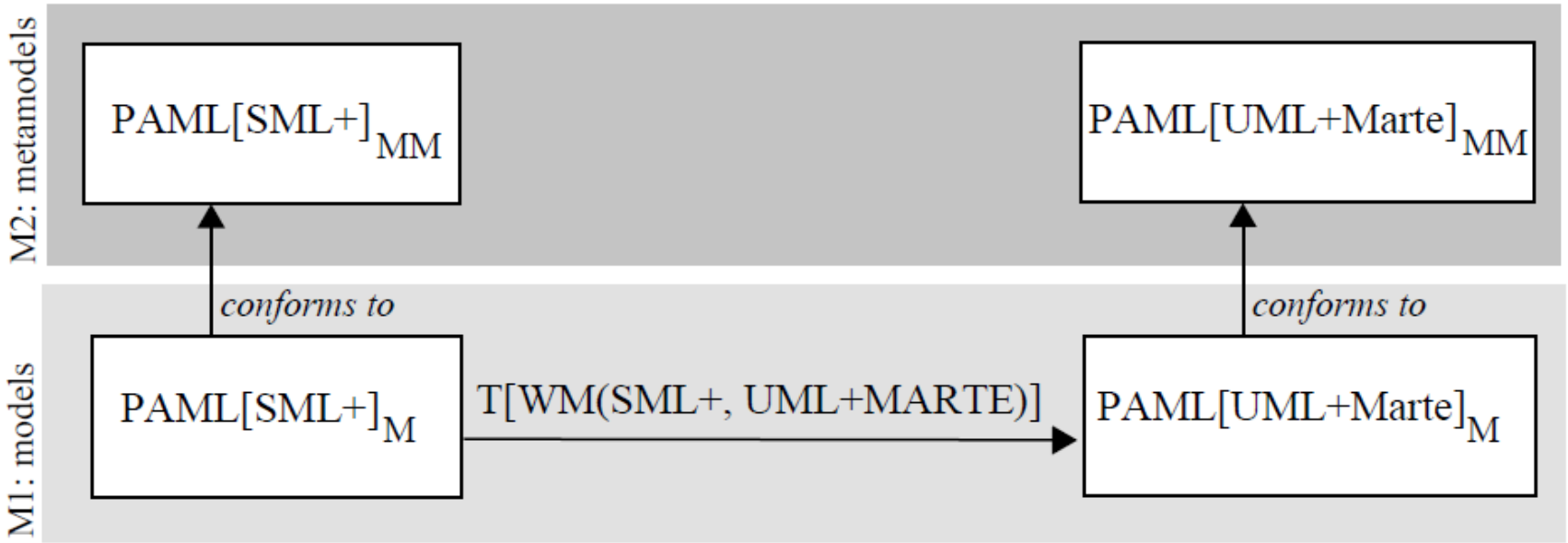
» Embedding Antipatterns neutral concept in concrete modeling notations

- Antipatterns are based on a dedicated general purpose language (SML+)
- Weaving models can map the concepts of SML+ into concrete modeling notations (parameter passing)



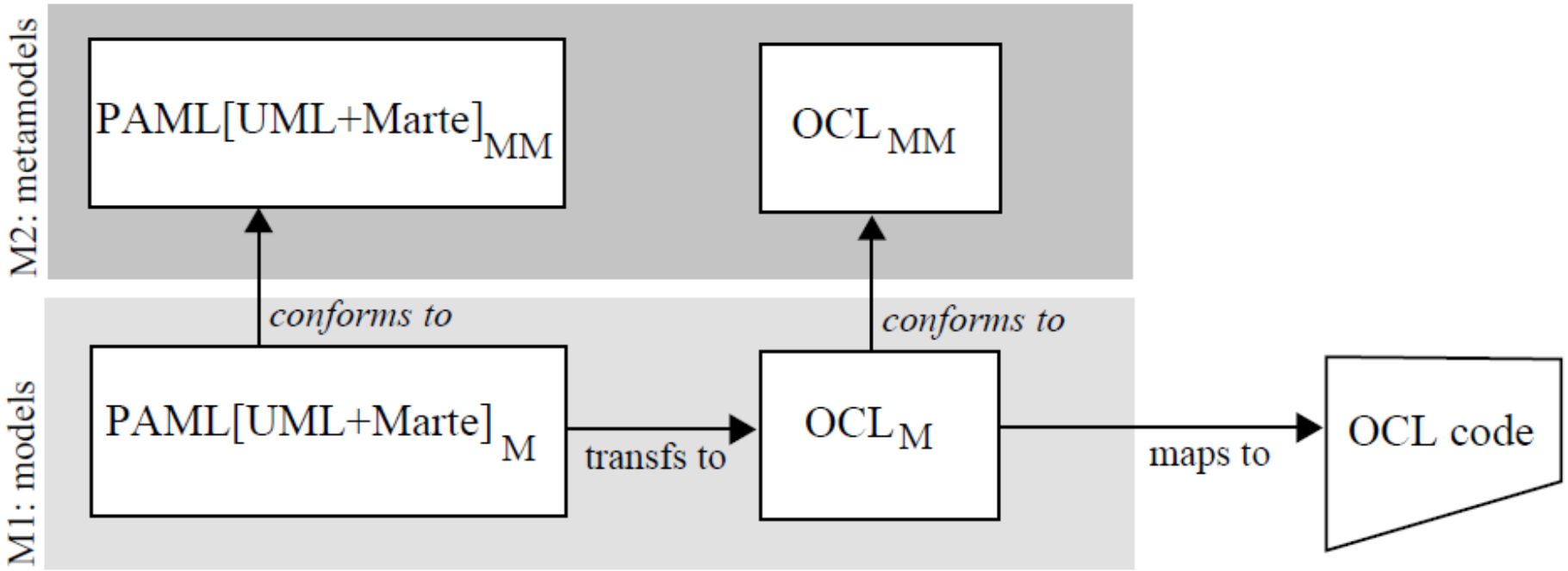
REPRESENTATION

A transformation T is generated, starting from the WM, between the neutral representation of antipatterns and the concrete modeling language



DETECTION

A transformation T' is generated, starting from the same WM, to provide an OCL-based executable semantics



Open Issues - Basics

- Gap textual/formal representations
- Threshold setting

- Uncertainty/incompleteness in :
  > Models, AP specifications, workload, op. profile

- Validation on larger model repositories (real case studies)

Open Issues - Advanced

- Evaluating the framework on:
  - > Precision (ratio of actual antipatterns found)
  - > Recall (ratio of antipatterns found overall)
- Language-specific antipatterns (other languages to experiment?)
- Antipattern at the model level and the code level

- Combining approaches (antipatterns with metaheuristics)
- Tool construction and integration

Open Issues – Process related

- Legacy constraints
- Conflicting solutions
- Overlapping solutions (composition)

- Combination with other quality attributes (maintenability, reliability, etc…)

We are obviously **open to collaborations** on any mentioned and non-mentioned point in this domain

Thank you!

{vittorio.cortellessa,
antinisca.dimarco,
catia.trubiani}@univaq.it