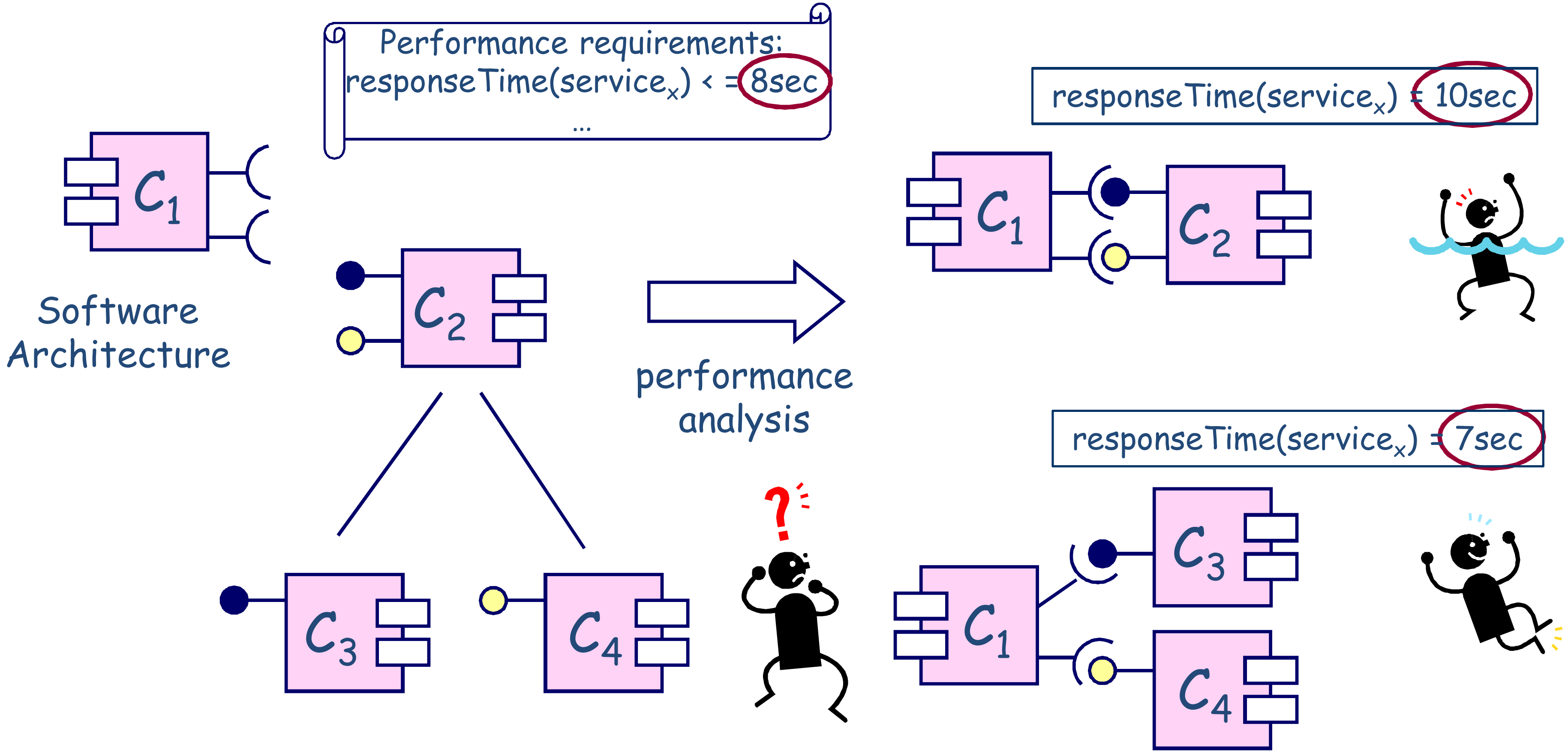
Enabling Performance Antipatterns to arise from an ADL-based Software Architecture

Vittorio Cortellessa, Martina De Sanctis,
Antinisca Di Marco, Catia Trubiani
University of L'Aquila
L'Aquila, Italy

Joint IEEE/IFIP Conference on Software Architecture
& European Conference on Software Architecture
Helsinki, Finland, 20-24 August 2012

» What to change (at the architectural level) in order to improve the software performance?



Performance requirements:
responseTime(service$_x$) < = 8sec
...

Software Architecture

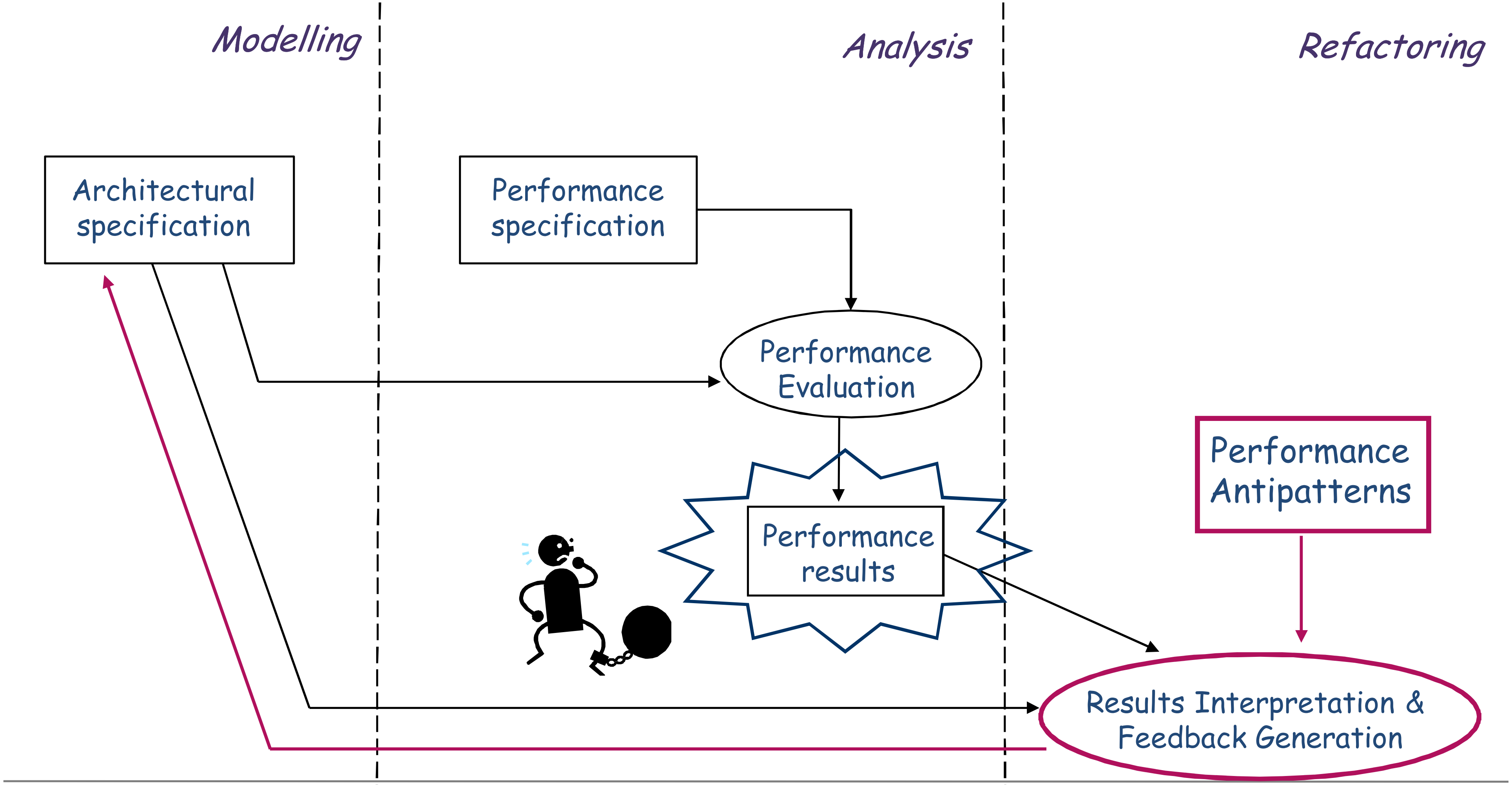
performance analysis

responseTime(service$_x$) = 10sec

responseTime(service$_x$) = 7sec

V. Cortellessa, M. De Sanctis, A. Di Marco, C. Trubiani: "Enabling Performance Antipatterns to arise from an ADL-based Software Architecture", WICSA/ECSA @ Helsinki, Finland, 20-24 August 2012

» Problem statement


Modelling
Analysis
Refactoring
Architectural specification
Performance specification
Performance Evaluation
Performance results
Performance Antipatterns
Results Interpretation & Feedback Generation

V. Cortellessa, M. De Sanctis, A. Di Marco, C. Trubiani: "Enabling Performance Antipatterns to arise from an ADL-based Software Architecture", WICSA/ECSA @ Helsinki, Finland, 20-24 August 2012

» **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

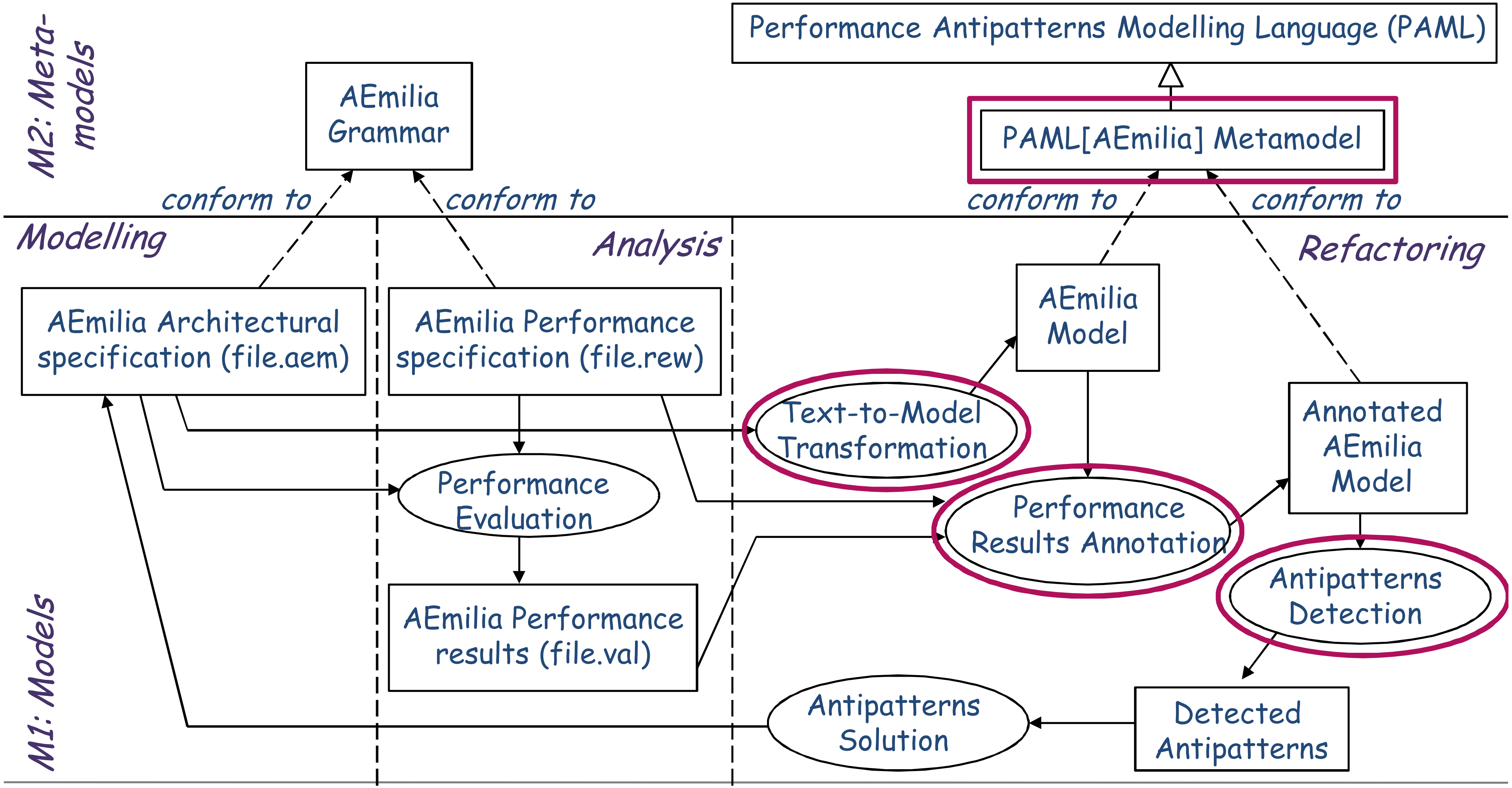
W.J.Brown, R.C. Malveau, H.W. Mc Cornich III, and T.J. Mowbray. "Antipatterns: Refactoring Software, Architectures, and Project in Crisis", 1998.

» <u>**Performance Antipatterns**:</u> what to avoid and how to solve performance problems

Antipattern		**Problem**	**Solution**
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.
...		...	...
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.

» Round-trip performance process in the AEmilia ADL



V. Cortellessa, M. De Sanctis, A. Di Marco, C. Trubiani: "Enabling Performance Antipatterns to arise from an ADL-based Software Architecture", WICSA/ECSA @ Helsinki, Finland, 20-24 August 2012

» Text-to-Model Transformation

AEmilia Grammar

conform to

AEmilia Architectural specification (file.aem)

```
   ...
ELEM_TYPE Sender_Type(const rate msg_gen_rate,
                      const rate timeout_rate)

   BEHAVIOR
    Sender_0(void; void) =
      <generate_msg, exp(msg_gen_rate)> .
      <transmit_msg_0, inf> . Sender_0_Waiting();

       ...

   INPUT_INTERACTIONS
     UNI generate_msg;
         receive_ack_0;
         receive_ack_1

   OUTPUT_INTERACTIONS
     UNI transmit_msg_0;
         transmit_msg_1
      ...
```

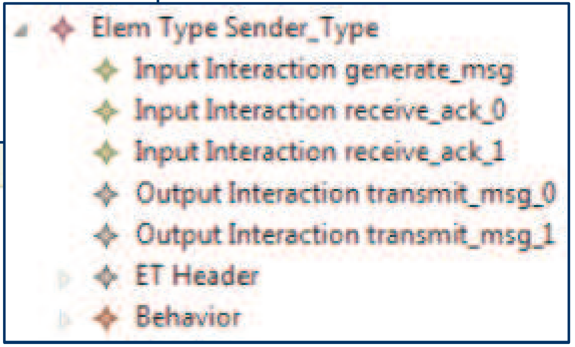
transformation rules

```
rule 'mapElemTypes'
      from elem_type et
      to ElemType
queries
      name : /#et;
      elemHeader : /et/#et_header;
      inputInt : /et//interaction_list_input
                 //#interactionInput;
      outputInt : /et//interaction_list_output
                 //#interactionOutput;
      behavior : /et/#behavior_equation_list;
mappings
      etName = name.WORD;
      elemHeader = elemHeader;
      iiDecl = inputInt;
      oiDecl = outputInt;
      behaviorDecl = behavior;
end_rule
```

PAML[AEmilia] Metamodel

conform to

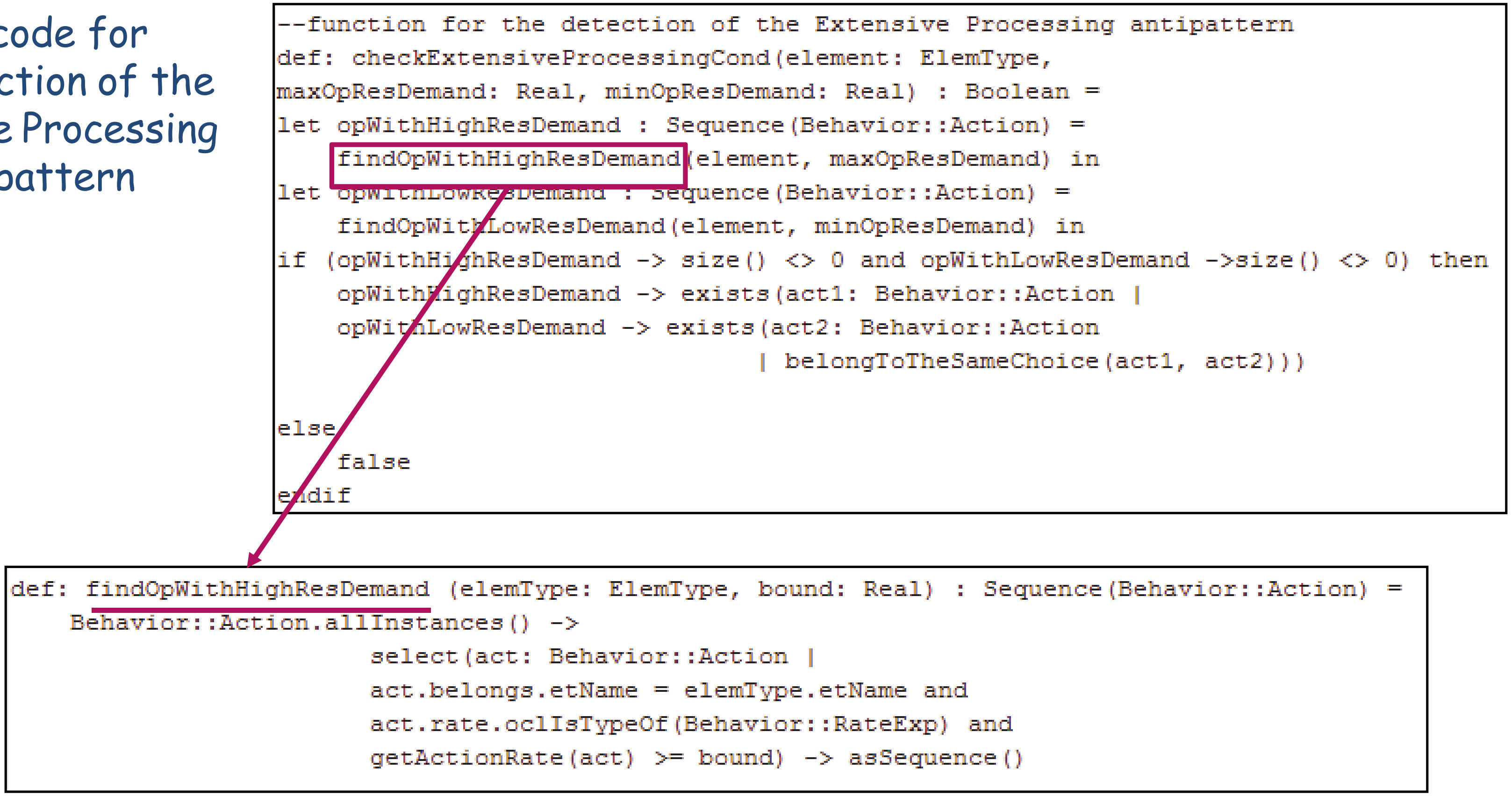
AEmilia Model



```
Elem Type Sender_Type
    Input Interaction generate_msg
    Input Interaction receive_ack_0
    Input Interaction receive_ack_1
    Output Interaction transmit_msg_0
    Output Interaction transmit_msg_1
    ET Header
    Behavior
```

» Antipatterns Detection

OCL code for the detection of the Extensive Processing Antipattern

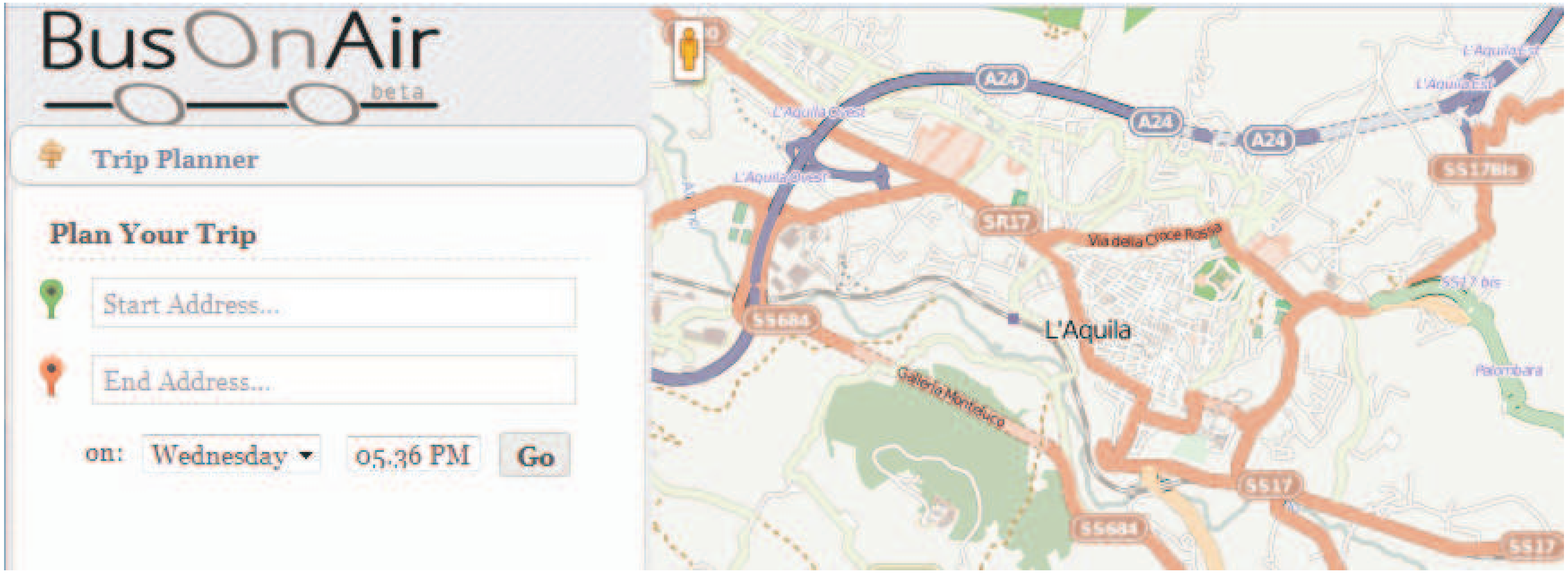


```
--function for the detection of the Extensive Processing antipattern
def: checkExtensiveProcessingCond(element: ElemType,
maxOpResDemand: Real, minOpResDemand: Real) : Boolean =
let opWithHighResDemand : Sequence(Behavior::Action) =
    findOpWithHighResDemand(element, maxOpResDemand) in
let opWithLowResDemand : 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()
```

» Bus on Air (BoA)

http://www.busonair.eu



Trip Planner

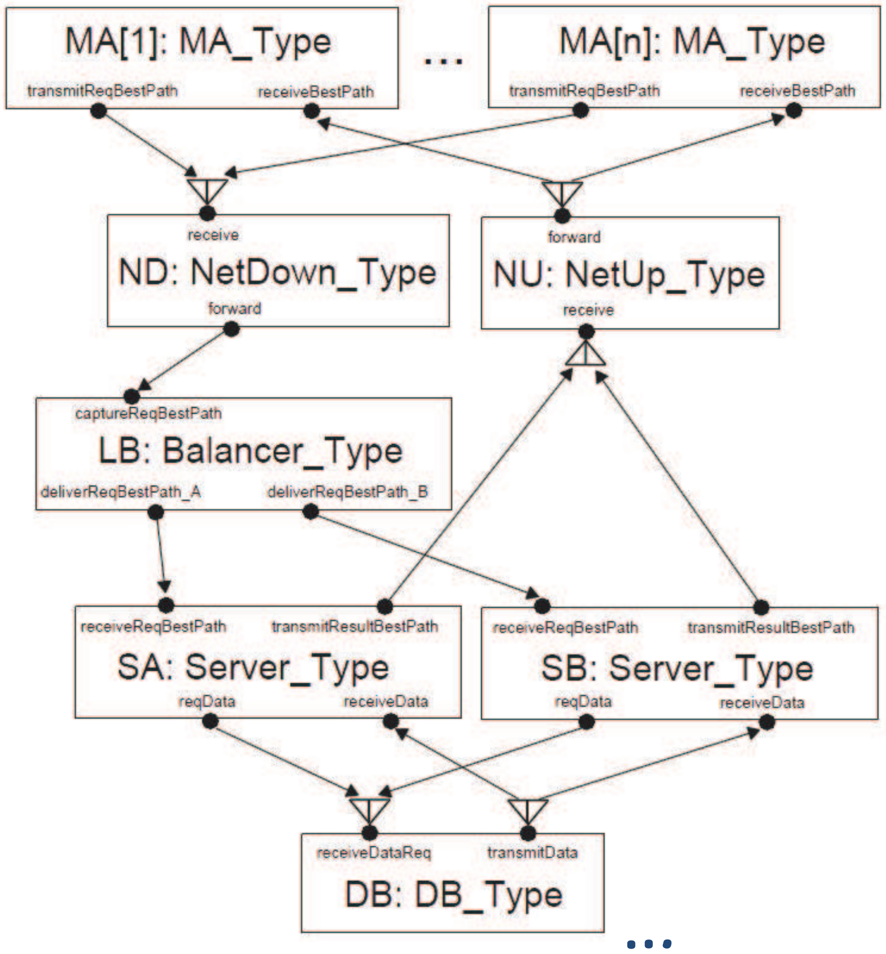
Plan Your Trip

Start Address...

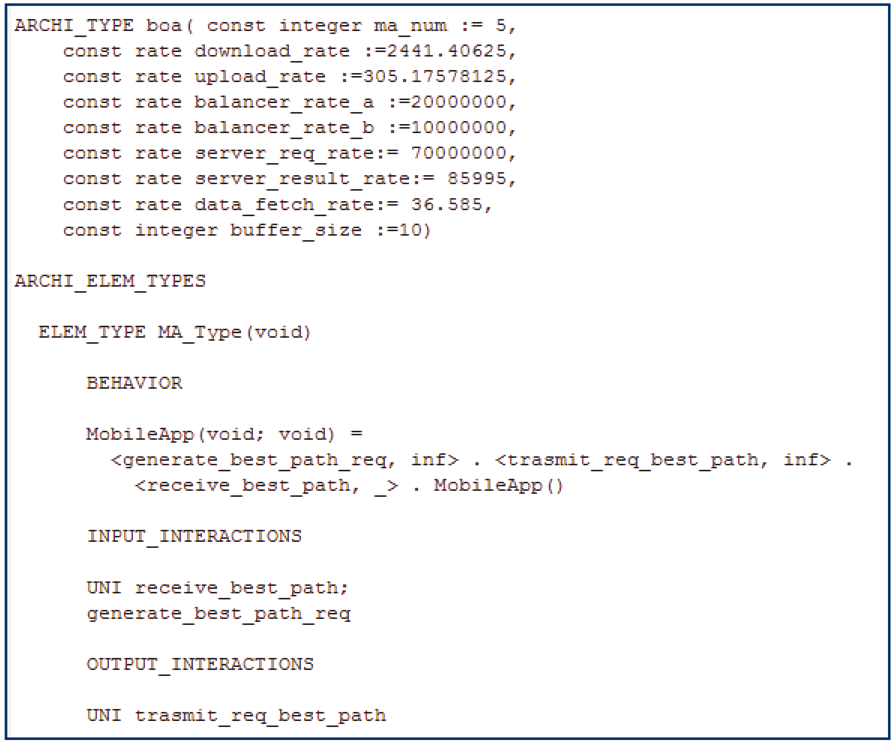
End Address...

on: Wednesday 05:36 PM Go

» Modeling

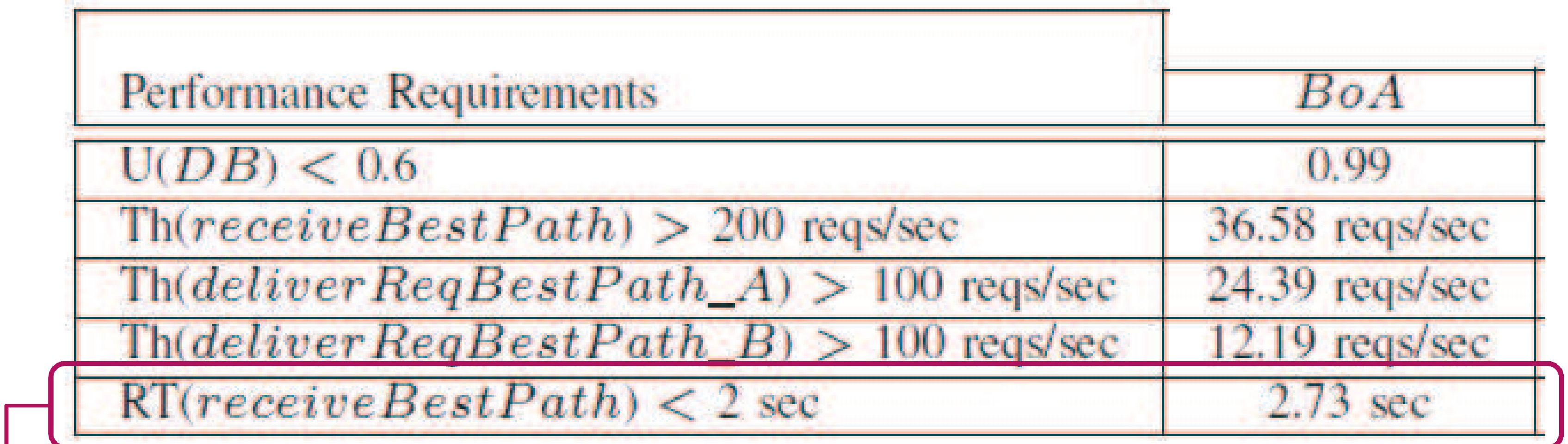


AEmilia Architectural specification (boa.aem)



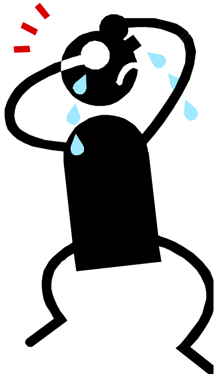
```
ARCHI_TYPE boa( const integer ma_num := 5,
    const rate download_rate :=2441.40625,
    const rate upload_rate :=305.17578125,
    const rate balancer_rate_a :=20000000,
    const rate balancer_rate_b :=10000000,
    const rate server_req_rate:= 70000000,
    const rate server_result_rate:= 85995,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

» Analysis

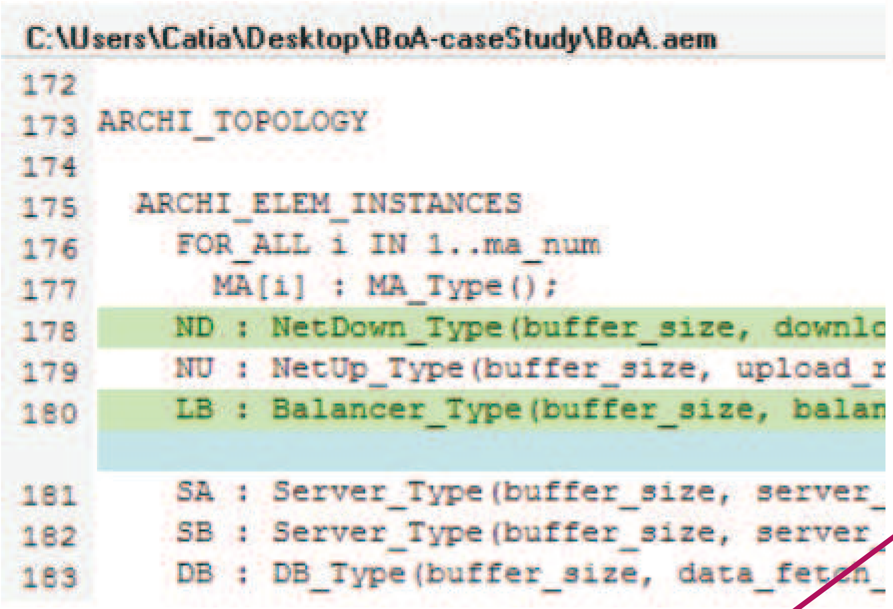


Performance Requirements	BoA
$\mathrm{U}(DB) < 0.6$	0.99
$\mathrm{Th}(receiveBestPath) > 200$ reqs/sec	36.58 reqs/sec
$\mathrm{Th}(deliverReqBestPath\_A) > 100$ reqs/sec	24.39 reqs/sec
$\mathrm{Th}(deliverReqBestPath\_B) > 100$ reqs/sec	12.19 reqs/sec
$\mathrm{RT}(receiveBestPath) < 2$ sec	2.73 sec

e.g. the user has to receive the best path with a RESPONSE TIME not larger than 2 seconds whereas the performance analysis predicts that the response time is equal to 2.73 seconds

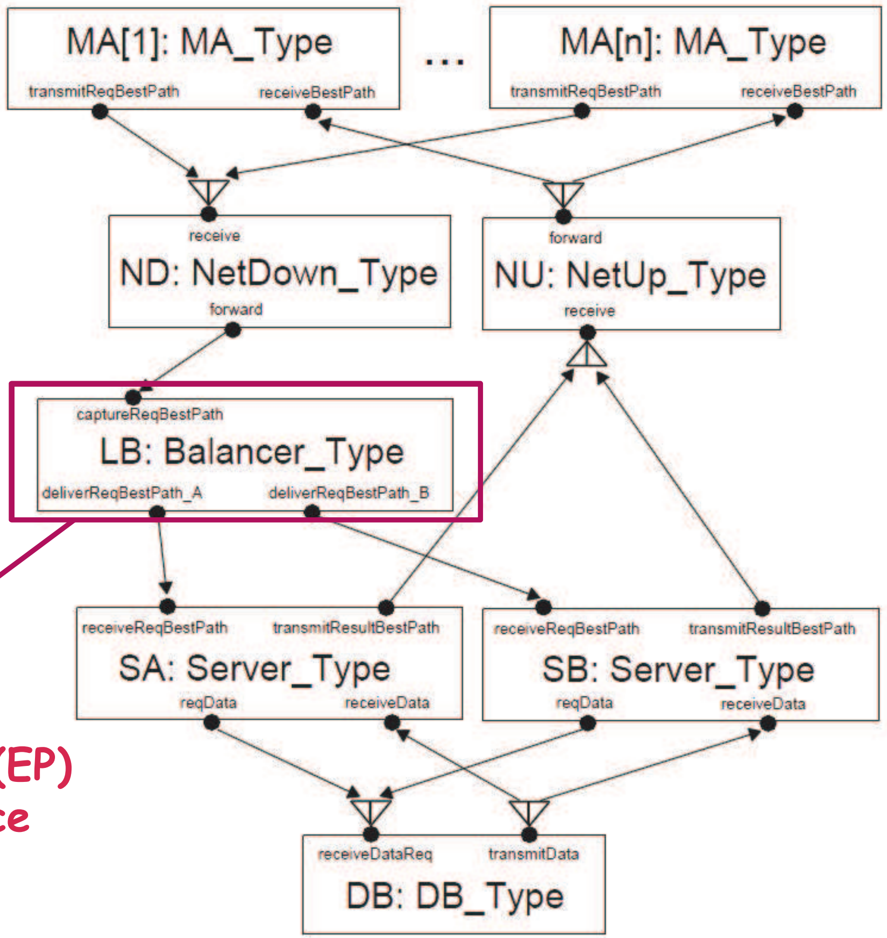


» Refactoring- detecting antipatterns



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

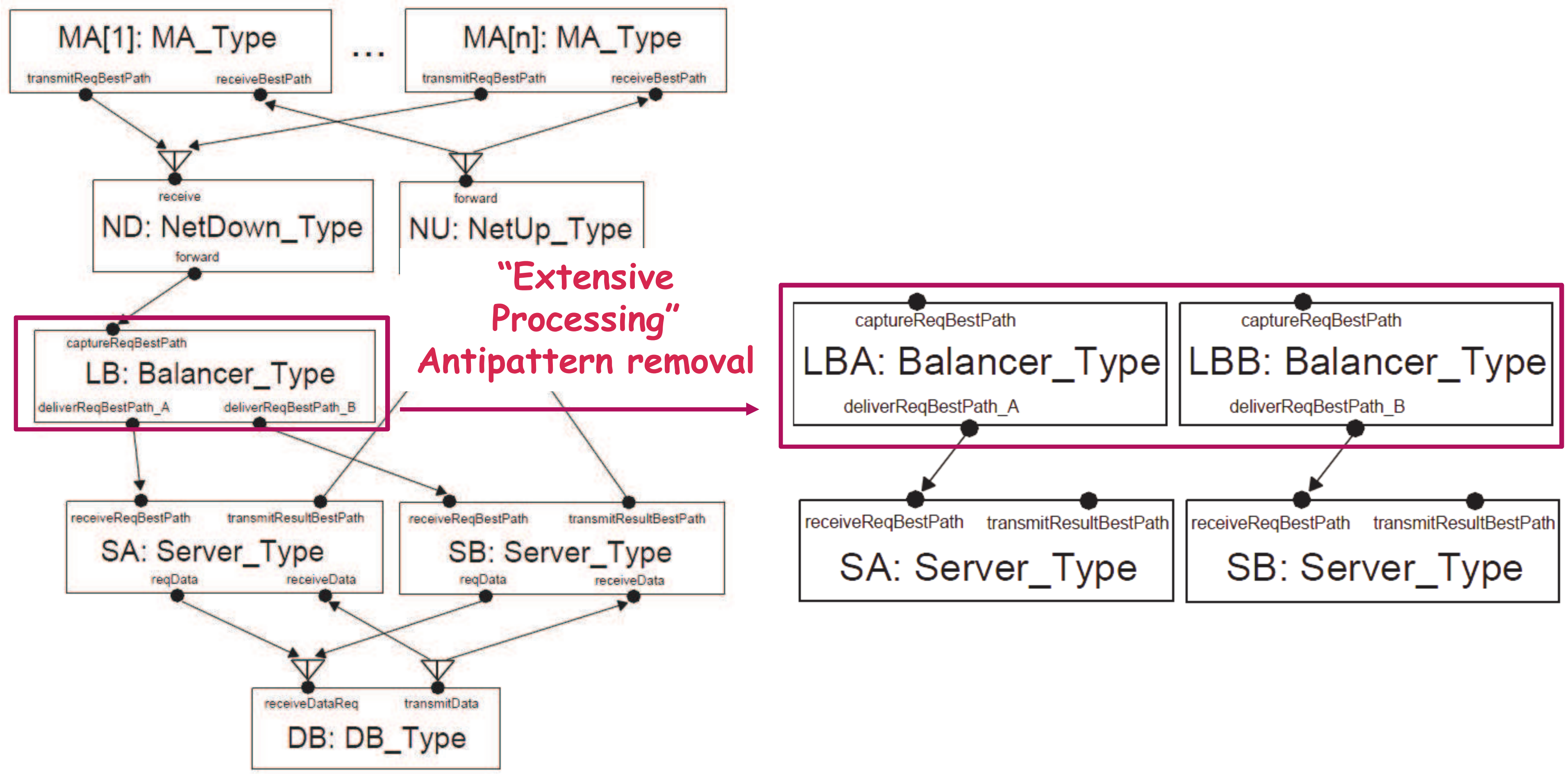
```
ARCHI_TOPOLOGY

  ARCHI_ELEM_INSTANCES
    FOR_ALL i IN 1..ma_num
      MA[i] : MA_Type();
    ND : NetDown_Type(buffer_size, downlo
    NU : NetUp_Type(buffer_size, upload_r
    LB : Balancer_Type(buffer_size, balan

    SA : Server_Type(buffer_size, server_
    SB : Server_Type(buffer_size, server_
    DB : DB_Type(buffer_size, data_fetch_
```



"Extensive Processing" (EP)
Antipattern occurrence

V. Cortellessa, M. De Sanctis, A. Di Marco, C. Trubiani: "Enabling Performance Antipatterns to arise from an ADL-based Software Architecture", WICSA/ECSA @ Helsinki, Finland, 20-24 August 2012

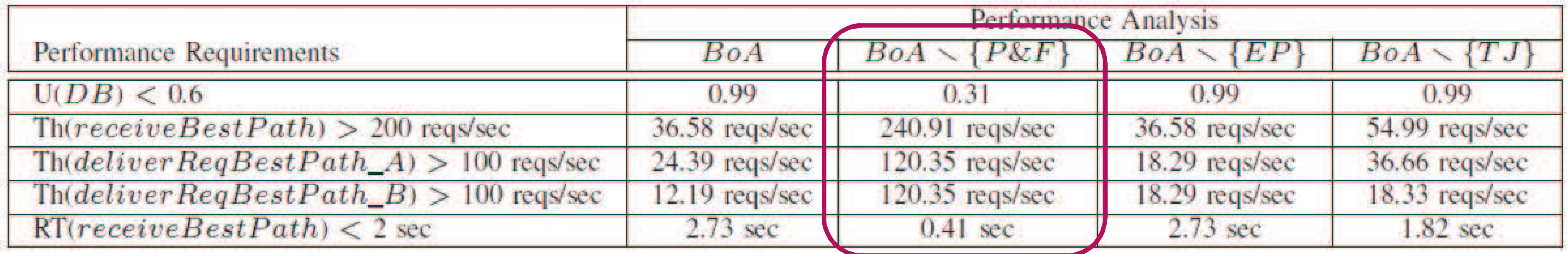
» Refactoring- solving antipatterns



"Extensive Processing" Antipattern removal

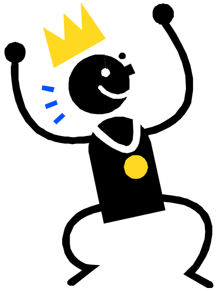
V. Cortellessa, M. De Sanctis, A. Di Marco, C. Trubiani: "Enabling Performance Antipatterns to arise from an ADL-based Software Architecture", WICSA/ECSA @ Helsinki, Finland, 20-24 August 2012

» Analysis of the refactored architectures



Performance Requirements	Performance Analysis			
	BoA	$BoA \setminus \{P\&F\}$	$BoA \setminus \{EP\}$	$BoA \setminus \{TJ\}$
U(DB) < 0.6	0.99	0.31	0.99	0.99
Th($receiveBestPath$) $>$ 200 reqs/sec	36.58 reqs/sec	240.91 reqs/sec	36.58 reqs/sec	54.99 reqs/sec
Th($deliverReqBestPath\_A$) $>$ 100 reqs/sec	24.39 reqs/sec	120.35 reqs/sec	18.29 reqs/sec	36.66 reqs/sec
Th($deliverReqBestPath\_B$) $>$ 100 reqs/sec	12.19 reqs/sec	120.35 reqs/sec	18.29 reqs/sec	18.33 reqs/sec
RT($receiveBestPath$) $<$ 2 sec	2.73 sec	0.41 sec	2.73 sec	1.82 sec

e.g. the removal of the "Pipe and Filter" (P&F) Antipattern allows to fulfill all the requirements



Contributions

- A model-driven approach to detect performance antipatterns in ADL-based software architectures
- Implementation of a tool that automatically detects antipatterns in AEmilia specifications

Future works

- Experimenting the approach on other ADLs (e.g. AADL, EAST-ADL, etc.)
- Validation of the approach on industrial case studies
- Introducing automation for antipatterns solution

Thank you!

For further information please refer to:

http://code.google.com/p/panda-aemilia

Questions

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