Model-Driven
Non-Functional Analysis

Vittorio Cortellessa
Dipartimento di Informatica
Università dell’Aquila
SUMMARY

- Non-functional analysis and software development
- Model-driven performance analysis
- Model-driven reliability analysis
- Model-driven tradeoff analysis
- Tool support
- Research perspectives
Non-functional analysis of software

Software modeling domain (vocabulary) is intrinsically distant from non-functional analysis one.

Reluctance to embed non-functional modeling and analysis in the software development process.
Non-functional analysis of software

“Human made” (ad-hoc) non-functional models: difficult to build and prone to errors
Non-functional analysis of software

A new vision in non-functional analysis (1998-today)

Introducing automation to generate non-functional models from software models

Big effort devoted to design model transformations having performance/reliability/... models as targets
Non-functional analysis of software: a general schema

Basic Software Model (original notation)

Validation of Functional Requirements

Ready-to-validate Non-Functional Model (usually new notation)

Validation of Non-Functional Requirements

Additional Information: software annotations

Missing data to be embedded into a software model for NF validation
Model annotations

In **UML** the **MARTE Specification** shall play the role of “container” of concepts related to non-functional attributes

*OMG, A UML Profile for MARTE, ptc/07-08-04*

All non-**UML** notations that intends to embed non-functional concepts have to define appropriate syntax and semantics for those concepts

In general, different non-functional attributes share more than one concept, thus it may be reasonable to design “core” specifications for common concepts plus specific concepts for each attribute
Non-functional analysis of software: an instance of the schema...

UML
Use Case Diagram, Sequence Diagram, Component Diagram

Queueing Network Model

Validation of Functional Requirements

Operational Profile, Resource Demand, ...

Performance Validation
Non-functional analysis of software: ... and yet another instance

UML:
Use Case Diagrams,
Sequence Diagrams,
Deployment Diagrams

Operational Profile,
Failure Probabilities, ...

Fault Tree Model

Validation of Functional Requirements

Reliability Validation
Non-functional models in Model-Driven Architecture

Horizontal and vertical transformations

Progetto PRIN PaCo (Performability-aware Computing)
Kickoff Meeting, Bertinoro, 23-24 ottobre 2008
Non-functional models in Model-Driven Architecture
Non-functional models in Model-Driven Architecture

- Systematic embedding of non-functional modeling and analysis in a model-driven development process
- Automation is the key to allow a real integration of these activities within the process
- The definition of CINFM, PINFM, PSNFM may not be trivial (or may be nonsense) for certain non-functional attributes
- The same consideration applies to the additional transformations introduced in NFMDA
SUMMARY

- Non-functional analysis and software development
- Model-driven performance analysis
- Model-driven reliability analysis
- Model-driven tradeoff analysis
- Tool support
- Research perspectives
Software Performance MDA: XPRIT methodology

Annotation and transformation
Software Performance MDA: CIM → CIPM

Performance requirements are annotated on a Use Case Diagram
Software Performance MDA: PIM → PIPM

Software dynamics is annotated with operational profile and resource demands and transformed in Execution Graphs.

At a limited extent, performance analysis can be carried out at Platform Independent level.
Software Performance MDA: PSM → PSPM

A Deployment Diagram is annotated with device characteristics and transformed into a Queuing Network representing the computational platform.

Workload is synthesized from the Execution Graphs (PIPM).

Performance analysis embedding resource contention is carried out at Platform Specific level.
SUMMARY

• Non-functional analysis and software development

• Model-driven performance analysis

• Model-driven reliability analysis

• Model-driven tradeoff analysis

• Tool support

• Research perspectives
Software Reliability MDA: COBRA methodology

Annotation and transformation
Software Reliability MDA:
\[ \text{CIM} \longrightarrow \text{CIRM} \]

Reliability requirements are annotated on a Use Case Diagram
Software Reliability MDA: PIM $\rightarrow$ PIRM

Software dynamics is annotated with **operational profile** and **probabilities of software component failures** and transformed into a function (of random variables)

Reliability analysis (only based on **software failures**) can be carried out at Platform Independent level
A Deployment Diagram is annotated with **hardware failure** (sites and connectors) **probabilities** and transformed in another function (of random variables)

Reliability analysis (based on **software and hardware failures**) can be carried out at Platform Specific level
SUMMARY

- Non-functional analysis and software development
- Model-driven performance analysis
- Model-driven reliability analysis
  - Model-driven tradeoff analysis
- Tool support
- Research perspectives
Model-driven tradeoff analysis

1. Optimization models for cost/quality tradeoffs

2. A compositional approach to security/performance tradeoff
Model-driven tradeoff analysis

1. Optimization models for cost/quality tradeoffs

2. A compositional approach to security/performance tradeoff
Search-Based Software Engineering

It is inspired by the observation that many activities in software engineering can be formulated as optimization problems.

Operational Researchers
- **Exact optimization techniques** of operation research, like linear or dynamic programming

Software Engineers
- **Metaheuristic search techniques**, like genetic algorithms, simulated annealing and tabu search, to find near optimal or good-enough solutions
The (class of) problem(s) faced

Studying the **tradeoffs between software cost and other non-functional attributes** (such as reliability and performance) in different phases of the software lifecycle

We have tackled this problem under the development paradigm of Component-Based Software Engineering in relationship to the selection of components.
Selection of components and lifecycle

Selection of components and lifecycle

- Human-based search

- Usually based only on functional characteristics of components

- Non-functional characteristics of components make the quality of a software product

- SBSE may help to raise the focus of software engineers from manual selection to model parameterization

- Efficient model solution techniques may allow to quickly analyze multiple alternatives
Selection of components and lifecycle

The problem

Selection of components driven by non-functional properties in Component-based Systems

Requirements Phase

Cost vs satisfaction of requirements

Architectural Design Phase

Cost vs reliability and delivery time

Implementation/Deployment Phase

Cost vs reliability and completion time
We have introduced an optimization model that determines the instance to choose for each component (either one of the available COTS products or an in-house developed one) in order to minimize the software costs under delivery time and reliability constraints.

In addition, the solution provides the amount of testing to be performed on each in-house component in order to achieve a certain reliability level.
Architectural Design Phase

We assume in this phase that the architecture of the software system has been designed

- All functionally equivalent
- They differ for cost and non-functional properties

For each component $i$:

- $\overline{J}_i$ is a set of instances that can be in-house developed
- $J_i$ is a set of COTS instances that can be bought
VARIABLES

In general, a “build-or-buy” decisional strategy can be described as a set of 0/1 variables defined as follows ($\forall i = 1, \ldots, n$):

$$x_{ij} = \begin{cases} 
    1 & \text{if instance } C_{ij} \text{ is chosen } (j \in J_i \text{ or } j \in \overline{J}_i) \\
    0 & \text{otherwise}
\end{cases}$$

The variables must fulfill the following constraints:

$$\sum_{j \in J_i \cup \overline{J}_i} x_{ij} = 1, \quad \forall i = 1, \ldots, n$$

For each component $i$, exactly one instance is either bought as COTS or in-house developed.
We express the Cost Objective Function as follows:

\[
\sum_{i=1}^{n} \left( \sum_{j \in J_i} \bar{C}_{ij} (t_{ij} + \tau_{ij} N_{ij}^{tot}) x_{ij} \right) + \sum_{j \in J_i} c_{ij} x_{ij}
\]

For each instance \(j\) and component \(i\) let:

- \(\bar{C}_{ij}\) be the unitary development cost
- \(t_{ij}\) be the estimated development time
- \(\tau_{ij}\) be the average time required to perform a test case
- Cost of an in-house instance: development cost plus testing cost
- \(N_{ij}^{tot}\) (additional variable): number of tests on in-house instance \(j\) of \(i\)
**Cost Objective Function**

We express the Cost Objective Function as follows:

$$\sum_{i=1}^{n} \left( \sum_{j \in J_i} c_{ij}(t_{ij} + \tau_{ij}N_{ij}^{tot})x_{ij} + \sum_{j \in J_i} c_{ij}x_{ii} \right)$$

Cost of a COTS component

For each instance $j$ and component $i$ let:

$c_{ij}$ be the buying cost
DELIVERY TIME CONSTRAINT

A maximum threshold $T$ is given on the delivery time of the whole system.

The following expression represents the delivery time of the component $i$:

$$\sum_{j \in J_i} (t_{ij} + \tau_{ij} N_{ij}^{tot}) x_{ij} + \sum_{j \in J_i} d_{ij} x_{ij}$$

Delivery time of an in-house instance.

For each instance $j$ and component $i$ let:

- $t_{ij}$ be the estimated development time
- $\tau_{ij}$ be the average time required to perform a test case
A maximum threshold $T$ is given on the delivery time of the whole system.

The following expression represents the delivery time of the component $i$:

$$\sum_{j \in J_i} (t_{ij} + \tau_{ij} N_{ij}^{tot}) x_{ij} + \sum_{j \in J_i} d_{ij} x_{ij}$$

For each instance $j$ and component $i$ let:

- $d_{ij}$ be the delivery time
A minimum threshold $R$ is given on the reliability of the whole system.

$\Phi$: non functional attribute (e.g., reliability)

Software architecture $SA$

$\Phi(SA) = \Phi(C1) \otimes \Phi(C2) \otimes ... \otimes \Phi(Cn)$

It is not easy to express the relationships between non-functional characteristics of components and quality of the whole system.
 RELIABILITY CONSTRAINT

\[ \text{REL(SA)} = \prod_{i=1}^{n} \Phi_i \]

\[ \Phi_i = e^{-fnum_i} \]

\[ fnum_i = \sum_{j \in J_i} \theta_{ij} s_i x_{ij} + \sum_{j \in J_i} \mu_{ij} s_i x_{ij} \]

The probability of failure on demand \( \theta_{ij} \) of the in-house developed instance \( C_{ij} \)

\[ \theta_{ij} = \frac{\text{Testab}_{ij} \ast p_{ij} (1 - \text{Testab}_{ij})^{N_{suc}^{ij}}}{(1 - p_{ij}) + p_{ij} (1 - \text{Testab}_{ij})^{N_{suc}^{ij}}} \]


the average number of failures of a component instance

the number of successful (i.e. failure free) tests performed on an in-house instance
**Optimization Model**

\[
\begin{align*}
\min & \quad \sum_{i=1}^{n} \left( \sum_{j \in J_i} c_{ij} (t_{ij} + \tau_{ij} N_{ij}^{\text{tot}}) x_{ij} + \sum_{j \in J_i} c_{ij} x_{ij} \right) \\
\max & \quad \left( \sum_{i=1}^{n} \sum_{j \in J_i} (t_{ij} + \tau_{ij} N_{ij}^{\text{tot}}) x_{ij} + \sum_{j \in J_i} d_{ij} x_{ij} \right) \leq T \\
\prod_{i=1}^{n} e^{-\left( \sum_{j \in J_i} \theta_{ij} s_{ij} x_{ij} + \sum_{j \in J_i} \mu_{ij} s_{ij} x_{ij} \right)} \geq R \\
\sum_{j \in J_i \cup \bar{J}_i} x_{ij} &= 1, \forall i = 1, \ldots, n \\
x_{ij} \in \{0,1\}, \forall i = 1, \ldots, n, \forall j \in J_i \cup \bar{J}_i
\end{align*}
\]
A framework for component selection

Framework architecture

Model builder

- UML software model (Component Diagram, Sequence Diagram)

Model solver (LINGO)

- Optimization model

- Operational Goals of the system
- Set of COTS and assemblies of components

Model results
- component selection
- amount of testing on in-house comp.

Model results
- component selection
- component deployment

Progetto PRIN PaCo (Performability-aware Computing)
Kickoff Meeting, Bertinoro, 23-24 ottobre 2008
An example: a distributed medical system

It is a client/server system, where the AE Client subsystem is connected via a network (Network subsystem) to the AE Server subsystem. The communication between the entities of the system is performed using Digital Imaging and Communication in Medicine (DICOM) standard, which is typically used, for example, for producing, processing and exchanging medical images.
An example: a distributed medical system

First Scenario
An example: a distributed medical system

Second Scenario
An example: a distributed medical system
An example: a distributed medical system

<table>
<thead>
<tr>
<th>Component name</th>
<th>COTS alternatives</th>
<th>Cost $c_{ij}$</th>
<th>Average delivery time $d_{ij}$</th>
<th>Average no. of invocations $s_i$</th>
<th>Prob. of fail. on demand $p_{ij}$</th>
</tr>
</thead>
<tbody>
<tr>
<td>$C_1$ - AE Client</td>
<td>$C_{11}$, $C_{12}$</td>
<td>14, 6</td>
<td>3, 3</td>
<td>1.9</td>
<td>0.001, 0.11</td>
</tr>
<tr>
<td>$C_2$ - DICOM UL Client</td>
<td>$C_{21}$, $C_{22}$</td>
<td>6, 12</td>
<td>4, 3</td>
<td>2.3</td>
<td>0.009, 0.001</td>
</tr>
<tr>
<td></td>
<td>$C_{23}$, $C_{24}$</td>
<td>14, 14</td>
<td>3, 3</td>
<td>0.0001</td>
<td></td>
</tr>
<tr>
<td>$C_3$ - Network</td>
<td>$C_{31}$, $C_{32}$</td>
<td>12, 14</td>
<td>2, 4</td>
<td>2.6</td>
<td>0.005, 0.0003, 0.0001</td>
</tr>
<tr>
<td></td>
<td>$C_{33}$, $C_{34}$</td>
<td>15, 11</td>
<td>7, 5</td>
<td>0.006, 0.0003</td>
<td></td>
</tr>
<tr>
<td>$C_4$ - DICOM UL Server</td>
<td>$C_{41}$, $C_{42}$</td>
<td>5, 10</td>
<td>4, 3</td>
<td>2.6</td>
<td>0.006, 0.0001, 0.0001</td>
</tr>
<tr>
<td>$C_5$ - AE Server</td>
<td>$C_{51}$, $C_{52}$</td>
<td>5, 10</td>
<td>3, 5</td>
<td>0.9</td>
<td>0.004, 0.0003</td>
</tr>
<tr>
<td></td>
<td>$C_{53}$, $C_{54}$</td>
<td>11, 11</td>
<td>5, 7</td>
<td>0.0001</td>
<td></td>
</tr>
</tbody>
</table>

Parameters for COTS instances

Parameters for in-house developed components

<table>
<thead>
<tr>
<th>Component name</th>
<th>Development Time $t_{i0}$</th>
<th>Testing Time $\tau_{i0}$</th>
<th>Unitary development cost $c_{i0}$</th>
<th>Faulty Probability $p_{i0}$</th>
<th>Testability $Testab_{i0}$</th>
</tr>
</thead>
<tbody>
<tr>
<td>$C_1$ - AE Client</td>
<td>6</td>
<td>0.007</td>
<td>1</td>
<td>0.8</td>
<td>0.006</td>
</tr>
<tr>
<td>$C_2$ - DICOM UL Client</td>
<td>6</td>
<td>0.007</td>
<td>1</td>
<td>0.8</td>
<td>0.009</td>
</tr>
<tr>
<td>$C_3$ - Network</td>
<td>6</td>
<td>0.007</td>
<td>1</td>
<td>0.3</td>
<td>0.006</td>
</tr>
<tr>
<td>$C_4$ - DICOM UL Server</td>
<td>3</td>
<td>0.007</td>
<td>1</td>
<td>0.5</td>
<td>0.009</td>
</tr>
<tr>
<td>$C_5$ - AE Server</td>
<td>4</td>
<td>0.007</td>
<td>1</td>
<td>0.8</td>
<td>0.009</td>
</tr>
</tbody>
</table>
We have solved the optimization model for multiple values of bounds $T$ and $R$. The former spans from 4 to 30 whereas the latter from 0.89 to 0.99.
### Model Solution

<table>
<thead>
<tr>
<th>$T=4$</th>
<th>$R=0.89$</th>
<th>$R=0.97$</th>
<th>$R=0.99$</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>$[C_{11}, C_{21}, C_{31}, (C_{40}, 0), (C_{50}, 0)]$</td>
<td>$[C_{11}, C_{21}, C_{32}, (C_{40}, 128), C_{51}]$</td>
<td>$[C_{11}, C_{23}, C_{32}, (C_{40}, 139), (C_{50}, 0)]$</td>
</tr>
<tr>
<td></td>
<td>SysRel=0.95539, Cost=39</td>
<td>SysRel=0.970000, Cost=42.896</td>
<td>SysRel=0.990000, Cost=49.973</td>
</tr>
</tbody>
</table>

| $T=6$ | $[C_{11}, (C_{20}, 0), (C_{30}, 0), (C_{40}, 0), (C_{50}, 0)]$ | $[C_{11}, C_{22}, (C_{30}, 0), (C_{40}, 44), (C_{50}, 80)]$ | $[C_{11}, C_{22}, C_{32}, (C_{40}, 112), (C_{50}, 137)]$ |
|       | SysRel=0.95915, Cost=33 | SysRel=0.970008, Cost=39.868 | SysRel=0.99, Cost=48.743 |

| $T=12$ | $[C_{11}, (C_{20}, 0), (C_{30}, 0), (C_{40}, 0), (C_{50}, 0)]$ | $[C_{11}, (C_{20}, 0), (C_{30}, 233), (C_{40}, 0), (C_{50}, 0)]$ | $[C_{11}, (C_{20}, 467), (C_{30}, 532), (C_{40}, 129), (C_{50}, 150)]$ |
|        | SysRel=0.95915, Cost=33 | SysRel=0.970002, Cost=34.631 | SysRel=0.99, Cost=41.946 |

| $T=30$ | $[C_{11}, (C_{20}, 0), (C_{30}, 0), (C_{40}, 0), (C_{50}, 0)]$ | $[C_{11}, (C_{20}, 0), (C_{30}, 233), (C_{40}, 0), (C_{50}, 0)]$ | $[C_{11}, (C_{20}, 488), (C_{30}, 459), (C_{40}, 148), (C_{50}, 164)]$ |
|        | SysRel=0.95915, Cost=33 | SysRel=0.970002, Cost=34.631 | SysRel=0.990002, Cost=41.813 |

**Optimal solution for all previous cases**

**(i.e. selection of components and number of tests)**
Model-driven tradeoff analysis

1. Optimization models for cost/quality tradeoffs

2. A compositional approach to security/performance tradeoff
A compositional approach to security/performance tradeoff

Do we have time for it now?
Experimentation: Application Model

Banking System

- Withdraw Funds
- Query Account
- Transfer Funds
- Validate PIN

ATM_1

ATM_2

ATM_n

Central Server
Experimentation:
Application Performance Model

Validate PIN
Withdraw Funds
Query Account
Transfer Funds
Experimentation: towards an Application Security Performance Model

...and which **Security Solutions**?

- Withdraw Funds -> *Availability*
- Query Account -> *Access Control*
- Transfer Funds -> *Data Confidentiality*
Experimentation: Application Security Performance Model

Transfer Funds
-> Data Confidentiality
= Seq(Encipherment, Routing Control)
Let us compare two configurations:

1. Banking System
   (i.e. the Application Performance Model)

2. Banking System with Security Services
   (i.e. the Application Security Performance Model)
Experimental results

- Throughput Analysis (in WF use case)

Withdraw Funds

rate of request arrivals to the system

throughput of rcvAckWF
Experimental results

Throughput Analysis (in QA use case)

rate of request arrivals to the system

Query
Account

throughput of rcvAckQA
Experimental results

- Throughput Analysis (in TF use case)

![Graph showing throughput of rcvAckTF vs rate of request arrivals to the system.](image)
Experimental results

- Throughput Analysis (in use cases)
  - Withdraw Funds
  - Transfer Funds
  - Query Account
SUMMARY

- Non-functional analysis and software development
- Model-driven performance analysis
- Model-driven reliability analysis
- Model-driven tradeoff analysis
- Tool support
- Research perspectives
Our “zoo” of tools

http://sealabtools.di.univaq.it/SeaLab/index.html

• **MOSQUITO**
  MOdel driven conStruction of QUeuIng neTwOrks

• **COBRA**
  COmponent-Based Reliability Analysis

• **DEER**
  DEcision support for componEnt-based softwaRe

• **WEASEL**
  Web sErvice for Analyzing queueing networkS with multiplE soLvers
... and other “animals” yet to come

- **SEAL**
  Software architEcture uml modeling of Aemilia specification tool

- **GARFIEL**
  Generation of ARchitectural Feedback from analysIs of pErformance resuLts
MOSQUITO is a tool that generates Queueing Network and Execution Graph models from annotated UML models according to the SAP·one and the Prima-UML methodologies.
MOSQUITO has a client/server architecture. A MOSQUITO downloadable plugin implements the client side that provides the functionality to invoke the web services provided on the server side at University of L’Aquila.
The extension point used for **MOSQUITO** interface is the Eclipse popup menu. After the creation of a model the user must only select the UML files (*.uml) and activate the desired transformation.
Annotation example in MagicDraw
**COBRA** is a tool that generates (and solves) reliability models for component-based software from annotated UML models.
The most recent reliability model we had as a target:
  • is based on failure probability of software components
  • includes error propagation
  • does not consider connector and platform failures

\[ err^{(k)}(i, j) \]: probability that the application reaches comp. \( j \) after \( k \) control transfers, starting from comp. \( i \), and \( j \) produces an erroneous output

\[
err^{(k)}(i, j) = p^{(k)}(i, j) \cdot if(j) + ep(j) \cdot (1 - if(j)) \sum_{h=0}^{C} err^{(k-1)}(i, h) p(h, j)
\]

\( e \): vector of the probabilities that the application (for each possible initial component) produces an erroneous output

\[
e = (I - Q)^{-1} \cdot F \cdot (I - Q \cdot R \cdot ((I - F))^{-1} \cdot c
\]
COBRA is an Eclipse plugin and provides a GUI to annotate UML models (at the moment imported from Visual Paradigm) and invoke the reliability model solution.
DEER is a tool embedding (3 types of) transformations from UML and non-UML models to optimization models for supporting software component selection.
WEASEL basically offers a Web Service that allows to solve Queueing Network models, specified in PMIF format, by invoking multiple external solution tools.
WEASEL

• Up to now WEASEL provides input to the following solvers: QNAP, SHARPE, PDQ, PEPSY, MQNA1/MQNA2, PMVA, MVA-QFP, OPENQN/CLOSEDQN.

• Its architecture is open to easily add the capability to interact with other solvers. WEASEL is fully customizable and extensible, as users can add new tools and capabilities.

• WEASEL follows a simple client-server approach to accomplish this scope, based on the web service standard SOAP protocol and WSDL specification language.
End users can develop their own web clients and implement them with any programming language that provides SOAP support.

Currently available WEASEL clients:
- PHP-based command line interface
- AJAX (Asynchronous Javascript AndXml) - based web interface
- Delphi-based Windows application (PMIF Editor)
WEASEL

PMIF Editor screenshot
SUMMARY

- Non-functional analysis and software development
- Model-driven performance analysis
- Model-driven reliability analysis
- Model-driven tradeoff analysis
- Tool support
- Research perspectives
Research perspectives

- Non-functional attributes may be (strictly) correlated, so sophisticated models to capture dependencies are needed
- Conflicts between non-functional and functional analysis results

- Automated interpretation of analysis results and feedback generation
- Keeping aligned software models and analysis models (in both directions)

- Introducing an unified representation of non-functional attributes to favor tool interoperability

- Role of pivot languages in all previous problems
A new direction:
Model synchronization and differences