Optimization models for non-functional requirements validation

Vittorio Cortellessa
Computer Science and Engineering Department
Università dell'Aquila - Italy

vittorio.cortellessa@univaq.it
http://www.di.univaq.it/cortelle/
Presentation roadmap

• Software non-functional requirements (NFR)
• Non-functional attribute (NFA) composition
• Optimization models
• Conclusions and perspectives
Software non-functional requirements

“Good enough” Non Functional Requirements (NFR) specification:

• **Quantification rather than qualification**

  The average response time of BrowseCatalog service must not be higher than 1.5 seconds...

  rather than

  **The BrowseCatalog service must be quick**

• **Workload specification**

  The average response time of BrowseCatalog service must not be higher than 1.5 seconds under a maximum workload of 50 requests/second
Software non-functional requirements

Non-functional requirements validation cannot be effectively carried out without these good practices.

But what is it expected from **NF validation** in general?

**Early** artifacts (e.g. models) in the lifecycle -
Quantitatively **compare** different software solutions vs requirements.

**Late** artifacts (e.g. code) in the lifecycle -
Estimate **realistic values** of NF attributes.
Non-functional attribute composition

1) On one NF attribute

2) On multiple NF attributes
Non-functional attribute composition

1) On one NF attribute
Expressing (possibly in a closed form) the whole system attribute in terms of component/service attributes

\[ \Phi(S) = \Phi(C_1) \otimes \Phi(C_2) \otimes \ldots \otimes \Phi(C_n) \]
Non-functional attribute composition

2) Across different NF attributes
Expressing (possibly in a closed form) the relationships/tradeoffs/dependencies among attributes

V. Cortellessa, Optimization models for non-functional requirement validation
CREST Workshop: Requirements and Test Optimization - UCL, February 11-12, 2013
Component-based Software

Choice of components driven by non-functional properties

We have introduced several optimization models...

**Requirements Phase**
Cost vs satisfaction of requirements

Based on satisfaction functions [Filkenstein et al.]

**Architectural Phase**
Cost vs reliability and delivery time

Closed-form of performance indices cannot capture waiting times on shared resources

**Deployment Phase**
Cost vs reliability and performance

Opening to evolving/adaptive systems where new requirements come after deployment

**Maintenance Phase**
Change management

V. Cortellessa, Optimization models for non-functional requirement validation
CREST Workshop : Requirements and Test Optimization - UCL, February 11-12, 2013
ARCHITECTURAL PHASE

On the basis of an architectural design, for each software component we assume to have different COTS available to buy or different in-house versions to build.

We also assume that all components are equivalent by a functional viewpoint.

We intend to determine the combination of available COTS products and in-house developed components that minimizes the software costs under delivery time and reliability constraints.

As a “side effect”, we provide the amount of testing to be performed on each in-house component in order to achieve the required reliability level.
**Optimization Model**

\[
\begin{aligned}
\min \quad & \sum_{i=1}^{n} \left( \sum_{j \in \overline{J}_i} c_{ij} (t_{ij} + \tau_{ij} N_{ij}^{\text{tot}}) x_{ij} + \sum_{j \in J_i} c_{ij} x_{ij} \right) \\
\max \quad & \left( \sum_{i=1}^{n} \sum_{j \in \overline{J}_i} (t_{ij} + \tau_{ij} N_{ij}^{\text{tot}}) x_{ij} + \sum_{j \in J_i} d_{ij} x_{ij} \right) \leq T \\
\prod_{i=1}^{n} & \left( -\left( \sum_{j \in J_i} \theta_{ij} s_i x_{ij} + \sum_{j \in \overline{J}_i} \mu_{ij} s_i x_{ij} \right) \right) \geq R \\
\sum_{j \in \overline{J}_i \cup J_i} & x_{ij} = 1, \forall i = 1, \ldots, n \\
x_{ij} & \in \{0, 1\}, \forall i = 1, \ldots, n, \forall j \in J_i \cup \overline{J}_i
\end{aligned}
\]
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:

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

Cost of a COTS component

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

\(c_{ij}\) be the buying cost
A maximum threshold $T$ has been given on the delivery time of the whole system.

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

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

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 $T$ has been given on the delivery time of the whole system.

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

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

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

$d_{ij}$ be the delivery time
A minimum reliability $R$ is required for the whole system.

A closed-form expression represents the reliability of the whole system:

$$\prod_{i=1}^{n} e^{-\left(\sum_{j \in J_i} \theta_{ij}s_ix_{ij} + \sum_{j \in J_i} \mu_{ij}s_ix_{ij}\right)} \geq R$$

Here is the requirement/testing joint point...
Probability that no failure occurs during the execution of the $i$-th component [Jung et al., 1999]:

$$\phi_i = e^{-\text{fnum}_i}$$

$$\text{fnum}_i = \sum_{j \in J_i} \theta_{ij} s_{ij} + \sum_{j \in J_i} \mu_{ij} s_{ij}$$

The probability of failure on demand $\theta_{ij}$ of the in-house developed instance $C_{ij}$ [Bertolino et al., 1996]:

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

$N_{suc}_{ij}$ the number of successful (i.e. failure free) tests performed on an in-house instance.
Just another (newer) example of closed-form reliability growth model that we are using now...

\[ \theta_i = \left( 1 - \frac{1 - \pi_i}{(1 - \pi_i) + \pi_i(1 - \pi_i)(1 - \pi_i)N_i^{tot}} \right) \]
An example: a distributed medical informatics 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 informatics system

First Scenario

Behavioral view
An example: a distributed medical informatics system

V. Cortellessa, Optimization models for non-functional requirement validation
CREST Workshop: Requirements and Test Optimization - UCL, February 11-12, 2013
An example: a distributed medical informatics system

Third Scenario

V. Cortellessa, Optimization models for non-functional requirement validation
CREST Workshop : Requirements and Test Optimization - UCL, February 11-12, 2013
### Parameters for COTS products

<table>
<thead>
<tr>
<th>Component name</th>
<th>Parameters for 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</td>
<td>3</td>
<td>1.9</td>
<td>0.001</td>
</tr>
<tr>
<td>( C_2 ) DICOM UL Client</td>
<td>( C_{21} ) ( C_{22} ) ( C_{23} )</td>
<td>6</td>
<td>4</td>
<td>2.3</td>
<td>0.009</td>
</tr>
<tr>
<td>( C_3 ) Network</td>
<td>( C_{31} ) ( C_{32} ) ( C_{33} )</td>
<td>12</td>
<td>2</td>
<td>2.6</td>
<td>0.005</td>
</tr>
<tr>
<td>( C_4 ) DICOM UL Server</td>
<td>( C_{41} ) ( C_{42} )</td>
<td>5</td>
<td>4</td>
<td>2.6</td>
<td>0.006</td>
</tr>
<tr>
<td>( C_5 ) AE Server</td>
<td>( C_{51} ) ( C_{52} ) ( C_{53} ) ( C_{54} )</td>
<td>5</td>
<td>3</td>
<td>0.9</td>
<td>0.004</td>
</tr>
</tbody>
</table>

### Parameters for in-house developed components

<table>
<thead>
<tr>
<th>Component name</th>
<th>Development Time ( t_{10} )</th>
<th>Testing Time ( \tau_{10} )</th>
<th>Unitary development cost ( c_{10} )</th>
<th>Faulty Probability ( p_{10} )</th>
<th>Testability ( Testab_{10} )</th>
</tr>
</thead>
<tbody>
<tr>
<td>( C_1 ) AE Client</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></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>0.006</td>
</tr>
<tr>
<td>( C_3 ) Network</td>
<td>6</td>
<td>0.007</td>
<td>1</td>
<td>0.8</td>
<td>0.009</td>
</tr>
<tr>
<td>( C_4 ) DICOM UL Server</td>
<td>3</td>
<td>0.007</td>
<td>1</td>
<td>0.3</td>
<td>0.006</td>
</tr>
<tr>
<td>( C_5 ) AE Server</td>
<td>4</td>
<td>0.007</td>
<td>1</td>
<td>0.5</td>
<td>0.009</td>
</tr>
</tbody>
</table>

V. Cortellessa, Optimization models for non-functional requirement validation  
CREST Workshop : Requirements and Test Optimization - UCL, February 11-12, 2013
Model Solutions

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.
Introducing stochastic programming

The reliability constraint only considers the average number of invocations of a component across different scenarios

\[
\prod_{i=1}^{n} e^{-\left(\sum_{j \in J_i} \theta_{ij} x_{ij} + \sum_{j \in J_i} \mu_{ij} s_{ij} x_{ij}\right)} \geq R
\]
Introducing stochastic programming

This does not avoid that, for some scenario, the system reliability can be lower than $R$

(But how was the reliability requirement specified?)

$$\prod_{i=1}^{n} e^{-\left(\sum_{j \in J_i} \theta_{ij}s_i x_{ij} + \sum_{j \in J_i} \mu_{ij}s_i x_{ij}\right)} \geq R$$

Different approaches can be taken to “fix” this “approximation”…

V. Cortellessa, Optimization models for non-functional requirement validation  
CREST Workshop: Requirements and Test Optimization - UCL, February 11-12, 2013
Introducing stochastic programming

We are working on a 2-stages programming approach

1) Find the optimal solution of the original problem:

\[
\min \sum_{i=1}^{n} (c_i(t_i + \tau_i N_i^{tot})y_i + \sum_{j=1}^{m} c_{ij}x_{ij})
\]

\[
\prod_{i=1}^{n} e^{-f_i} \geq R \quad \text{where} \quad f_i = s_i(\lambda_i y_i + \sum_{j=1}^{m} \mu_{ij} x_{ij})
\]

\[
y_i + \sum_{j=1}^{m} x_{ij} = 1, \quad \forall i = 1 \ldots n
\]
Introducing stochastic programming

2) Then try to take “recourse actions” to compensate for possible inconsistencies in the original problem

\[
\min \left( \sum_{i=1}^{n} c_i \hat{t}_i y_i + \sum_{j=1}^{m} c_{ij} x_{ij} \right) + \left( \sum_{i=1}^{n} C_i \delta_i y_i + \sum_{j=1}^{m} C_{ij} \delta_{ij} x_{ij} \right) + \sum_{l=1}^{N} p^l \sum_{i=1}^{n} (C^l_i \delta^l_i)
\]

\[
\prod_{i=1}^{n} e^{-f^l_i} \geq R \quad \forall l = 1, \ldots, N
\]

\[
y_i + \sum_{j=1}^{m} x_{ij} = 1 \quad \forall i = 1 \ldots n
\]

\[
\delta_i \leq \lambda_i, \delta_{ij} \leq \mu_{ij} \quad \forall i = 1 \ldots n, \forall j = 1 \ldots m
\]

\[
0 \leq f^l_i \quad \forall i = 1 \ldots n, \forall l = 1, \ldots, N
\]

where \[
\hat{f}^l_i = (s^l_i - \delta^l_i)((\lambda_i - \delta_i)y_i + \sum_{j=1}^{m} (\mu_{ij} - \delta_{ij})x_{ij})
\]
We have provided the CODER (Cost Optimization under DElivery and Reliability constraints) tool, which generates and solves the optimization model.
Conclusions

We have introduced several optimization models for non-functional requirement validation

**PROS**

- Easy representation of modular software (e.g. component-based, service-oriented)
- Flexibility in the definition of cost functions
- Limited solution time for small/medium size problems (i.e. about 20 components and 10 instances for each component)
- Easy exploration of multiple alternatives
- **Capability of embedding stochastic parameters**
Conclusions

We have introduced several optimization models for nonfunctional requirement validation

**CONS**

- Only closed-form expressions can be adopted, and they do not capture all relevant aspects of non-functional attributes
- Exponential solution time for growing size problems (possibly mitigated with meta-heuristic approaches)
- ...
- Borderline research topic between Optimization and Software Engineering -> not the highest acceptance rate of papers 😊
Future perspectives

• Application on real large scale problems (metaheuristics)

• Stochastic optimization

• Optimization models as a support to runtime decisions (need quick-and-dirty solution approaches)

• ...
Acknowledgments

The ideas and results presented in this talk have been achieved in collaboration with:

Pasqualina Potena, Università di Bergamo
Fabrizio Marinelli, Università Politecnica delle Marche

Extra-credits also go to Pasqualina for sharing with me her large repository of slides 😊
Some references...

• Vittorio Cortellessa, Raffaela Mirandola, Pasqualina Potena: Selecting Optimal Maintenance Plans Based on Cost/Reliability Tradeoffs for Software Subject to Structural and Behavioral Changes. CSMR 2010:21-30


• Vittorio Cortellessa, Ivica Crnkovic, Fabrizio Marinelli, Pasqualina Potena: Driving the selection of cots components on the basis of system requirements. ASE 2007:413-416

• Vittorio Cortellessa, Fabrizio Marinelli, Pasqualina Potena: Automated Selection of Software Components Based on Cost/Reliability Tradeoff. EWSA 2006:66-81