dipartimentoinformatica
Università degli Studi dell'Aquila

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:

- **<u>Quantification rather than qualification</u>**

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

rather than

The BrowseCatalog service must be quick

- **<u>Workload specification</u>**

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

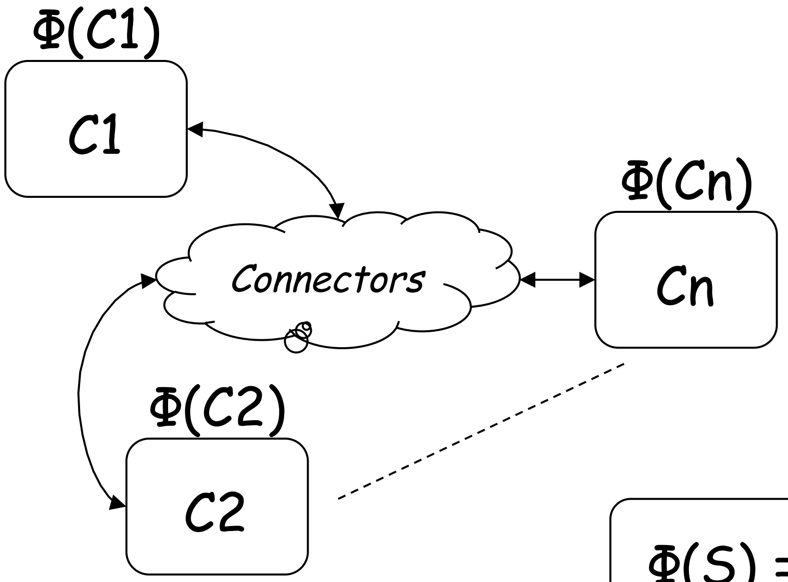
1) On one NF attribute

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



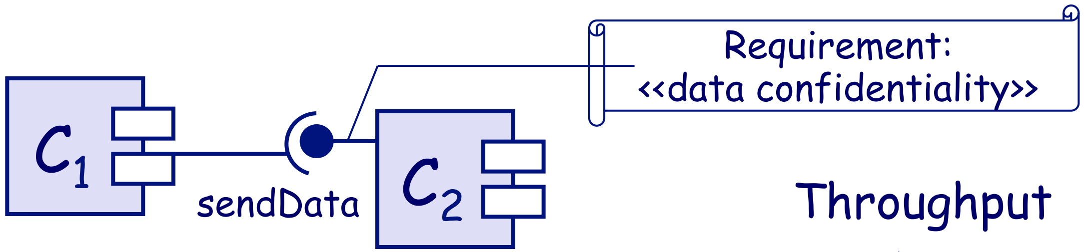
Φ() : non functional attribute (e.g., reliability)

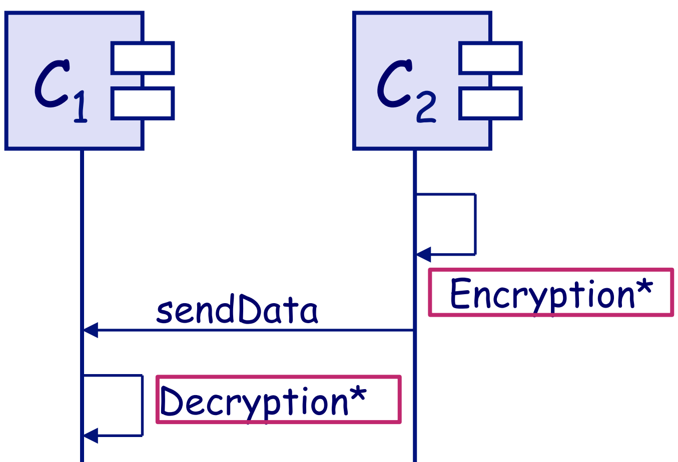
$$\Phi(S) = \Phi(C1) \otimes \Phi(C2) \otimes \ldots \otimes \Phi(Cn)$$

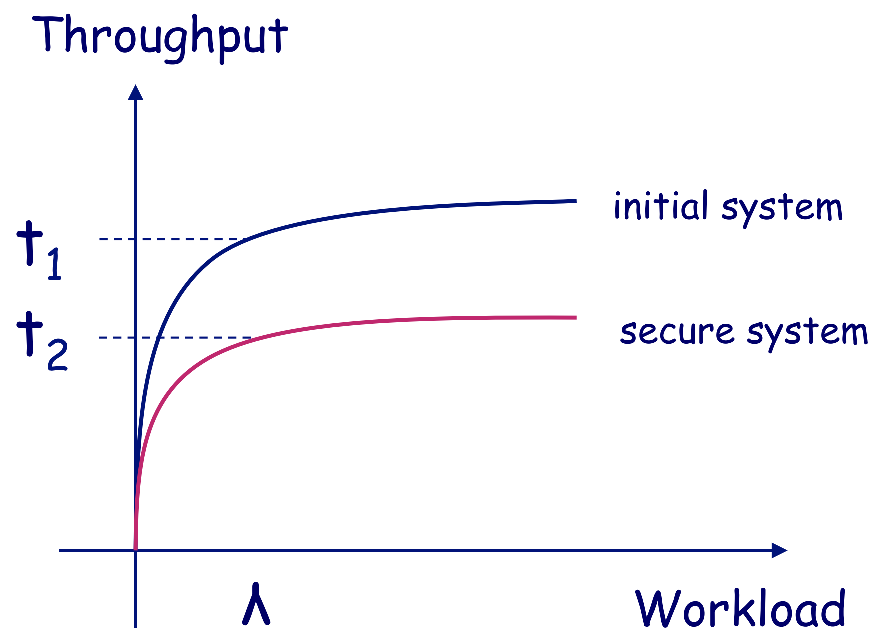
Non-functional attribute composition

2) Across different NF attributes

Expressing (possibly in a closed form) the relationships/tradeoffs/dependencies among attributes







Optimization models

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

DEPLOYMENT PHASE

Cost vs reliability and performance

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

MAINTENANCE PHASE

Change management

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

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

$$\min \sum_{i=1}^{n}\left(\sum_{j\in \bar{J}_i} \bar{c}_{ij}(t_{ij}+\tau_{ij}N_{ij}^{tot})x_{ij} + \sum_{j\in J_i} c_{ij}x_{ij}\right)$$

$$\max_{i=1,..,n}\left(\sum_{j\in \bar{J}_i}(t_{ij}+\tau_{ij}N_{ij}^{tot})x_{ij} + \sum_{j\in J_i} d_{ij}x_{ij}\right) \le T$$

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

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

$$x_{ij}\in\{0,1\}, \forall i = 1,\ldots,n, \forall j \in J_i \cup \bar{J}_i$$

Variables

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

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

The variables must fulfill the following constraints:

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

For each component *i*, exactly one instance is either bought as COTS or in-house developed.

Cost Objective Function

We express the Cost Objective Function as follows:

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

Cost of a COTS component

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

c_{ij} be the buying cost

Delivery Time Constraint

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*:

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

Delivery time of an in-house instance.

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

t_{ij} be the estimated development time

τ_{ij} be the average time required to perform a test case

DELIVERY TIME CONSTRAINT

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*:

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

Delivery time of a COTS component

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

d_{ij} be the delivery time

RELIABILITY CONSTRAINT

A minimum reliability *R* is required for the whole system.

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

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

Here is the requirement/testing joint point…

RELIABILITY CONSTRAINT

Probability that no failure occurs during the execution of the *i*-th component [Jung et al., 1999] :

$$\boxed{\phi_i = e^{-fnum_i}}$$

$$\boxed{fnum_i = \sum_{j \in \bar{J}_i} \theta_{ij} s_i x_{ij} + \sum_{j \in J_i} \mu_{ij} s_i x_{ij}}$$

average number of failures of a component instance

probability of failure on-demand

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

$$\boxed{\theta_{ij} = \frac{Testab_{ij} * p_{ij}(1-Testab_{ij})^{N_{ij}^{suc}}}{(1-p_{ij}) + p_{ij}(1-Testab_{ij})^{N_{ij}^{suc}}}}$$

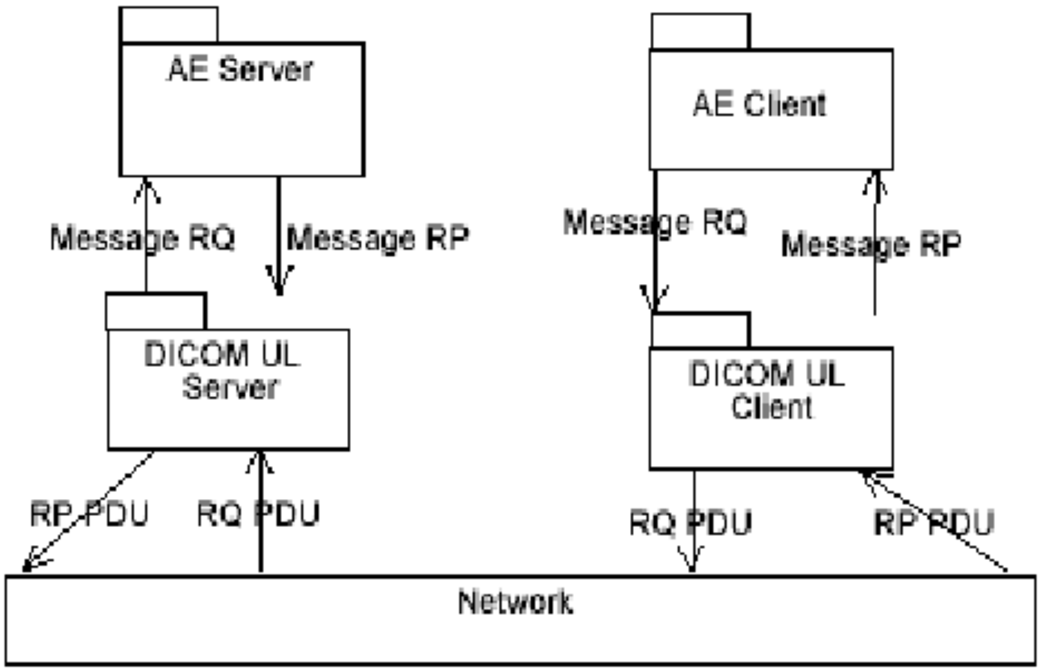
(other reliability growth models in closed forms can be adopted here)

N^{succ}_{ij} the number of succesful (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

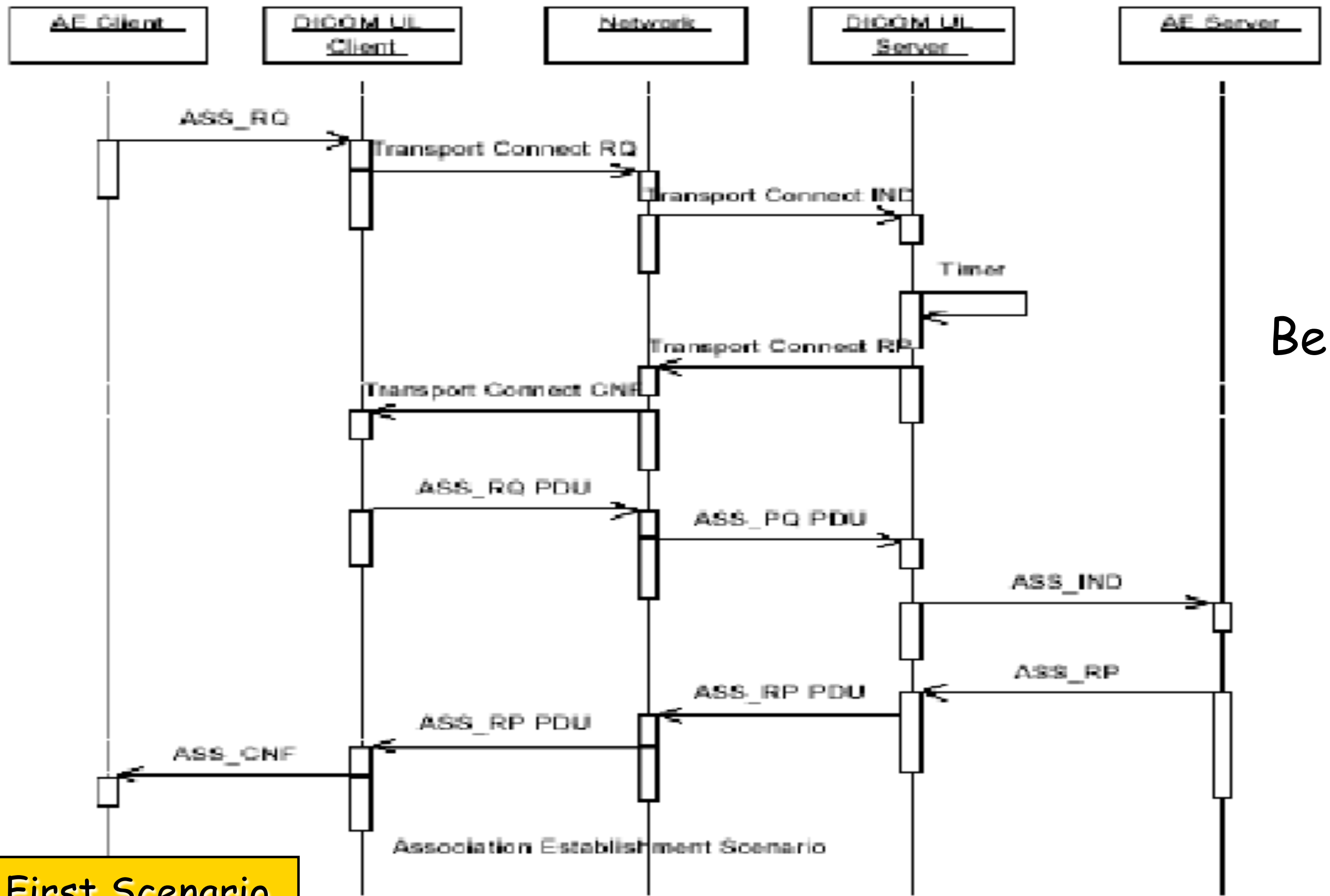


Static view

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.

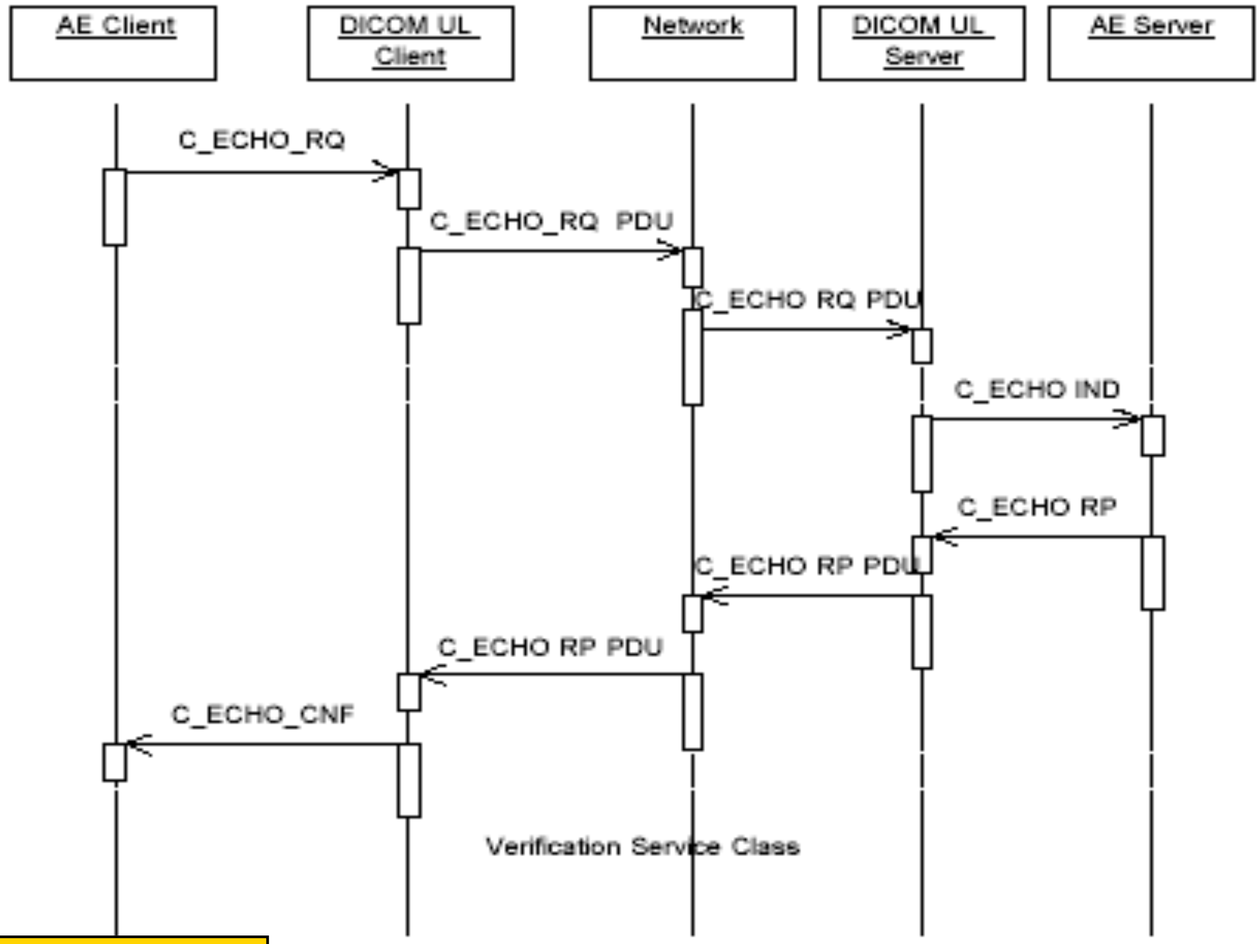
An example: a distributed medical informatics system



Behavioral view

First Scenario

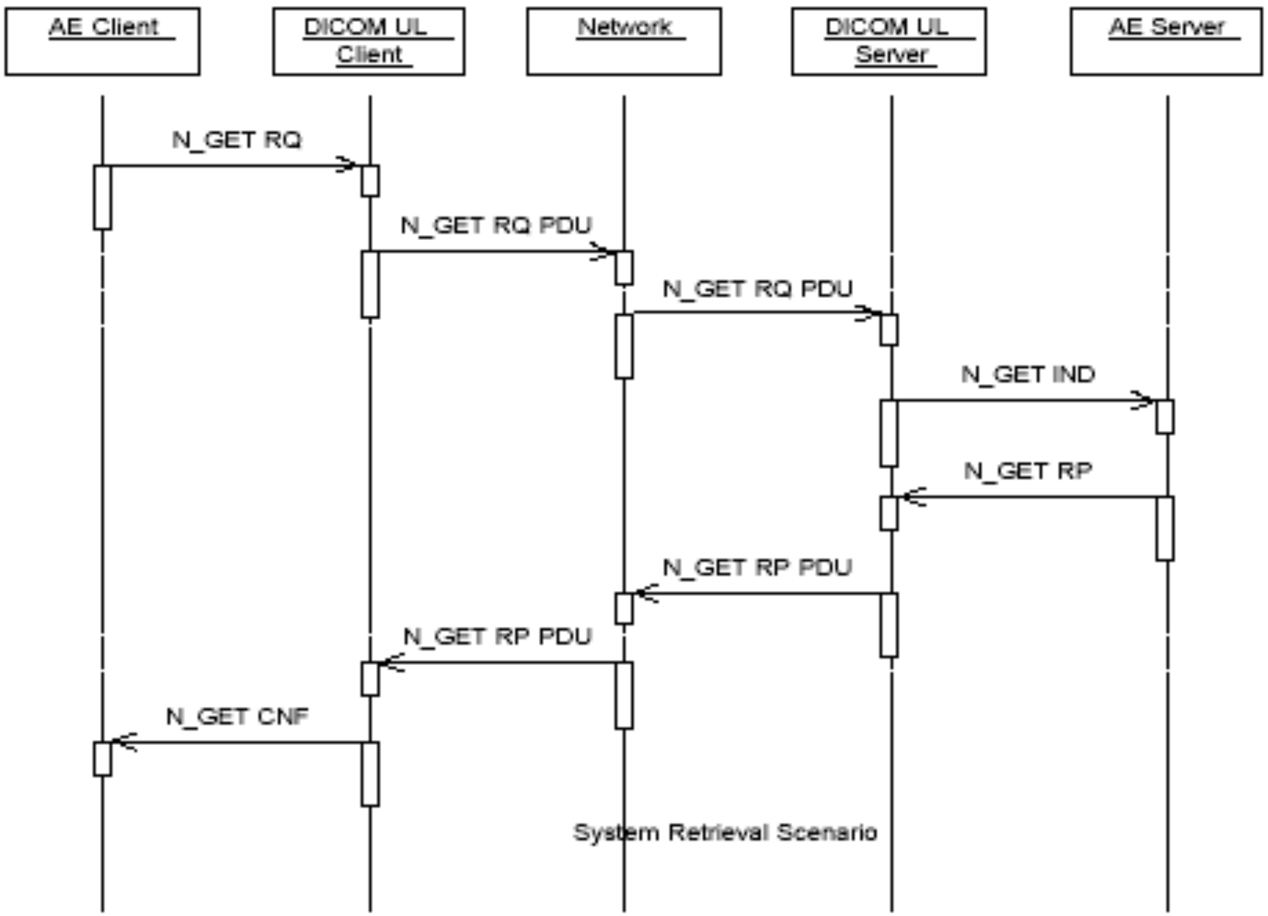
An example: a distributed medical informatics system



Behavioral view

Second Scenario

An example: a distributed medical informatics system



Behavioral view

Third Scenario

	Component name	COTS alternatives	Cost c_{ij}	Average delivery time d_{ij}	Average no. of invocations s_i	Prob. of fail. on demand μ_{ij}
C_1	*AE Client*	C_{11}	14	3	$1.\bar{9}$	0.001
		C_{12}	6	3		0.11
C_2	*DICOM UL Client*	C_{21}	6	4	$2.\bar{3}$	0.009
		C_{22}	12	3		0.001
		C_{23}	14	3		0.0001
C_3	*Network*	C_{31}	12	2	$2.\bar{6}$	0.005
		C_{32}	14	4		0.0003
		C_{33}	15	7		0.0001
C_4	*DICOM UL Server*	C_{41}	5	4	$2.\bar{6}$	0.006
		C_{42}	10	3		0.0002
C_5	*AE Server*	C_{51}	5	3	$0.\bar{9}$	0.004
		C_{52}	10	5		0.0003
		C_{53}	11	5		0.000001
		C_{54}	11	7		0.0001

Parameters for COTS products

	Component name	Development Time t_{i0}	Testing Time τ_{i0}	Unitary development cost $\bar{c}_{i0}$	Faulty Probability p_{i0}	Testability $Testab_{i0}$
C_1	*AE Client*					
C_2	*DICOM UL Client*	6	0.007	1	0.8	0.006
C_3	*Network*	6	0.007	1	0.8	0.009
C_4	*DICOM UL Server*	3	0.007	1	0.3	0.006
C_5	*AE Server*	4	0.007	1	0.5	0.009

Parameters for in-house developed components

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

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

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”…

Introducing stochastic programming

We are working on a 2-stages programming approach

1) Find the optimal solution of the original problem:

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

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

$$y_i + \sum_{j=1}^{m} x_{ij} = 1, \ \ \forall i = 1 \ldots n$$

Introducing stochastic programming

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

$$\min(\sum_{i=1}^{n} c_i \hat{t}_i y_i + \sum_{j=1}^{m} c_{ij} x_{ij}) + (\sum_{i=1}^{n} C_i \delta_i y_i + \sum_{j=1}^{m} C_{ij} \delta_{ij} x_{ij}) + \sum_{l=1}^{N} p^l \sum_{i=1}^{n} (C_i^l \delta_i^l)$$

$$\prod_{i=1}^{n} e^{-\hat{f}_i^l} \geq R \qquad \forall l = 1, \ldots, N$$

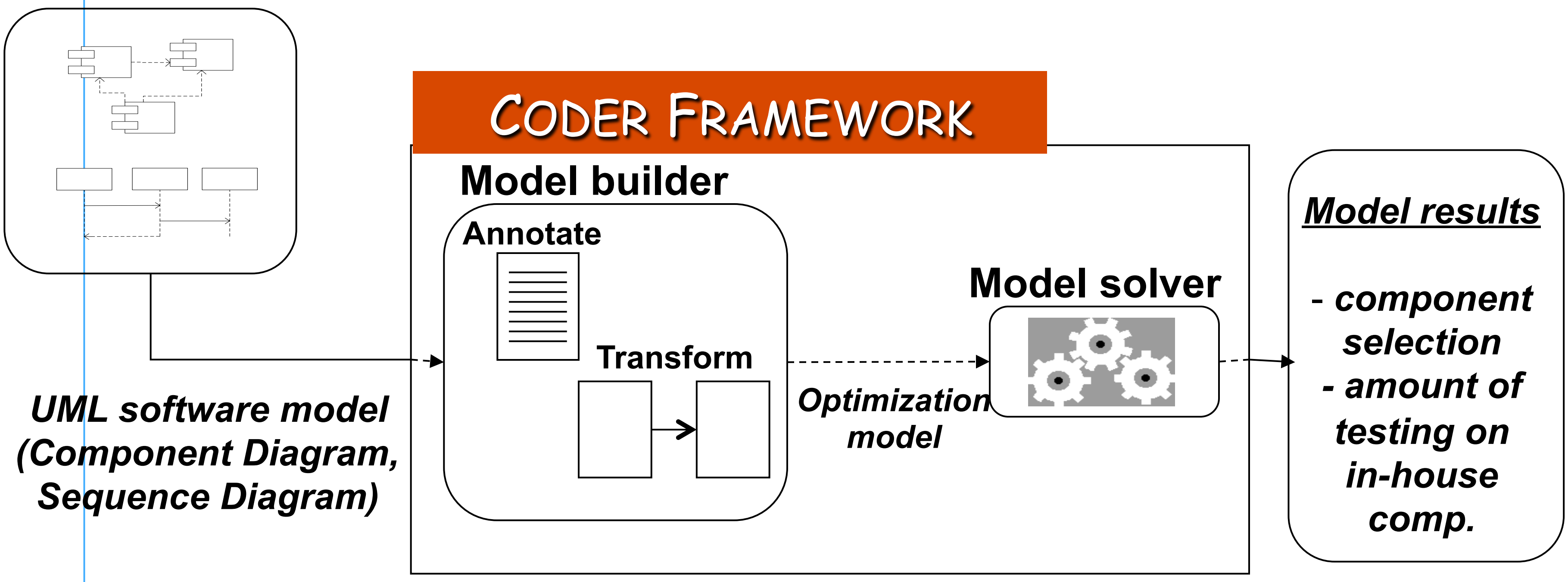
$$y_i + \sum_{j=1}^{m} x_{ij} = 1 \qquad \forall i = 1 \ldots n$$

$$\delta_i \leq \lambda_i, \delta_{ij} \leq \mu_{ij} \qquad \forall i = 1 \ldots n, \forall j = 1 \ldots m$$

$$0 \leq \hat{f}_i^l \qquad \forall i = 1 \ldots n, \forall l = 1, \ldots, N$$

where $\hat{f}_i^l = (s_i^l - \delta_i^l)((\lambda_i - \delta_i) y_i + \sum_{j=1}^{m} (\mu_{ij} - \delta_{ij}) x_{ij})$

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 non-functional 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 approahes)
- …

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, Fabrizio Marinelli, Pasqualina Potena: An optimization framework for "build-or-buy" decisions in software architecture. Computers & OR (COR) 35(10):3090-3106 (2008)

- Vittorio Cortellessa, Ivica Crnkovic, Fabrizio Marinelli, Pasqualina Potena: Experimenting the Automated Selection of COTS Components Based on Cost and System Requirements. J. UCS (JUCS) 14(8):1228-1255 (2008)

- 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