Handling non-functional requirements in Information System Architecture Design

Anargyros Tsadimas, Mara Nikolaidou, Dimosthenis Anagnostopoulos
Department of Informatics & Telematics
Harokopio University of Athens
70 El. Venizelou Str, 176 71 Athens, Greece
{*tsadimas, mara, dimosthe*}*@hua.gr*

Abstract

Information system architecture design is a complex task depending on both functional and non-functional requirements. Since system architecture definition is strongly related to system performance, non-functional requirements play a significant role during enterprise information system design. To explore the effect of non-functional requirements on system design process, a model-based approach emphasizing non-functional requirements is proposed. To facilitate the designer to effectively define and handle requirements during architecture design, a number of system views are proposed, each of them focusing on discrete design issues and satisfying different kind of requirements. A consistent requirement model is defined representing how non-functional requirements are related between them and to system components forming the overall system architecture. SysML has been adopted as the modeling language, since it enables requirement definition and can be formally extended. Moreover, requirement derivation process is discussed and a case study where the proposed concepts are applied in practice while redesigning the legacy system of a large-scale organization is presented.

1. Introduction

According to INCOSE [1], determining the system architecture (i.e. the way autonomous system components should be synthesized) is a complex process, which in essence focuses on the integration of system components already defined by other stakeholders than system architects. Thus, system design should be explored taking into account related requirements identified by the respective stakeholders.

A requirement denotes a capability or condition that must (or should) be satisfied and may specify a function that a system must perform or a condition a system must achieve [2]. Requirements are divided into two main categories: *functional* and *non-functional* [3] [4]. Non-functional requirements (NFRs) is a broadly used term, while there are significant efforts on how to handle them [5]; however, there is no consensus about their nature, since various classifications exist in the literature [3] [4]. NFRs play a significant role during system design, since they depict the conditions under which specific system components should operate, leading to alternative design decisions [6]. We suggest that the basic aspects of NFRs for system design purposes can be depicted in three sub-categories, namely *performance*, *constraint* and *specific quality* in accordance with other researchers [7]. During system design emphasis is often laid upon performance requirements [8].

Model-based system engineering is about elevating models in the engineering process to a central and governing role in the specification, design, integration, validation, and operation of a system. In such a case, activities that support the engineering process are to be accomplished by developing models of increasing detail [9] [10]. Model-based system engineering is supported by a number of methodologies and modeling languages [11]. In such a case, a central system model must be defined capturing all system requirements and decisions that fulfill them at different levels of abstraction, serving discrete activities and facilitating information exchange between them.

Enterprise Architecture (EA) frameworks [12] are characterized as an attempt to integrate strategies, processes, methods, models and tools towards enterprise information system engineering. Most of them have adopted the notion of separating concerns by establishing different viewpoints, each depicting the concerns of a specific stakeholder (e.g. user, designer, implementer, etc.) In [13], the concept of using Zachman EA framework as the basis for establishing a central EIS model for model-based EIS engineering was introduced. Zachman framework provides a holistic model of enterprise information infrastructure, focusing on 6 different perspectives and 6 different aspects. A plethora of methodologies and formalisms exist [14], [15], each applicable to a subset of cells of the Zachman matrix, while respective system models are defined. As such, the matrix may integrate different concerns, issues and methods, while specific methods may use parts of it as a reference point.

Based on the proposed concepts in [13], a model-based approach for the design of EIS architecture emphasizing NFRs is discussed in this paper. EIS architecture design is the process of defining and optimizing the architecture of the information system, both hardware and software, and exploring performance requirements. While application design mainly focuses on functional requirements, architecture design targets the effective EIS operation emphasizing on NFRs.

A model-based design approach could be integrated within Zachman EA matrix and allow for the progressive refinement of EIS architecture based on a well-defined model constituting of discrete *EIS Architecture Views* emphasizing different EIS architecture aspects (e.g. software architecture, hardware configuration, system topology). EIS Architecture model should facilitate the description of both functional and NFRs and the design decisions related to them according to different EIS Architecture perspectives often influenced by different EIS stakeholders, other than system designer. Each of the perspectives results to an independent EIS Architecture view serving a discrete design activity. Adopting a model-based approach for EIS Architecture design allows for the progressive and independent execution of these activities in parallel, while the impact of design decisions adopted in each of them to the others is expressed in terms of relations between corresponding EIS Architecture views.

Although there is a UML profile for Modeling Software Quality of Service [16], it is not adequent for system engineering. Thus, we chose SysML to support the model-based approach to EIS Architecture design. SysML [11] supports the concepts of requirement definition and management and resource allocation, which are vital to depict EIS Architecture design activities. A SysML profile can be defined to provide an integrated EIS Architecture model consisting of alternative system views addressing functional and NFRs . The profile was implemented as a plugin to MagicDraw modeling tool [17].

The rest of the paper is organized as follows: Section 2 explains the main concepts of model-based EIS architecture design identifying basic design activities and corresponding EIS views. Section 3 emphasizes on NFR management. In section 4, the corresponsing SysML profile is described. In section 5, a case study is presented, discussing the application of the proposed concepts in the renovation of a public organization legacy system. Conclusions and future work are discussed in section 6.

2. Model-Based Design of Enterprise Information System Architecture

EIS Architecture design is the process of defining EIS structure to effectively support EIS provided functionality. This activity should be performed taking into account related requirements identified by the respective stakeholders (application designers, managers, system engineers) having different views of EIS. In practice, EIS Architecture design consists of the definition and optimization of a system architecture comprised of software and hardware components, ensuring that all software components are identified and properly allocated and that hardware components are properly combined to support the efficient operation of software components. It is evident that the identification of functional requirements, e.g. EIS components and their capabilities [2], is not enough to ensure EIS efficient operation. NFRs, e.g the conditions under which EIS components should operate [2], should also be taken into account.

A model-based approach for EIS Architecture design is based on a common system model facilitating the definition of both functional and non-functional EIS requirements and the synthesis of a system architecture combining them [10]. Functional requirements and corresponding design decisions are described using complementary EIS Architecture views focusing on different aspects of system design. *Functional view* depicts functional requirements related to software components and relevant data, system users and design decisions related to software architecture. *Topology view* facilitates the description of system access points in terms of hierarchically related locations, called *sites* and deals with software allocation decisions. *Network Infrastructure view* refers to the aggregate network, described through a hierarchical structure comprising of simple and composite networks and defines hardware components and configurations. Related functional requirements are depicted in corresponding hardware component models, while network and hardware design decisions are explored. Non-functional requirements are represented using an independent EIS view, integrated in EIS Architecture model, called *NFR View* focused on performance requirements. NFRs defined within NFR view must be satisfied by specific entities included in Functional, Topology and Network Infrastructure views depicting specific EIS functionality and EIS Architecture design decisions. In this manner, the system architect is enabled to realize the affect of specific design decisions (for example the allocation of software to hardware resource) to NFRs imposed on them (for example performance) and vise-versa. All NFRs are aggregated in NFR view, while each of them is also included in the corresponding diagram that satisfies it. Using NFR view, the system designer is enabled to explore NFR relationships, while, using other views, the relationship between NFRs and design decisions is explored. Thus, it is important to provide a well-defined meta-model describing requirements and the relations between them and other EIS Architecture components [18].

Functional view (figure 1) encompasses functional specifications (e.g. application architecture, user behavior and data structures). Applications are considered to be based on multi-tiered, client-server models. Each application tier, called *module*, contains *services*. The behavior of different user groups is modeled through *roles*. *Data entities* are defined to indicate portions of data used by applications.

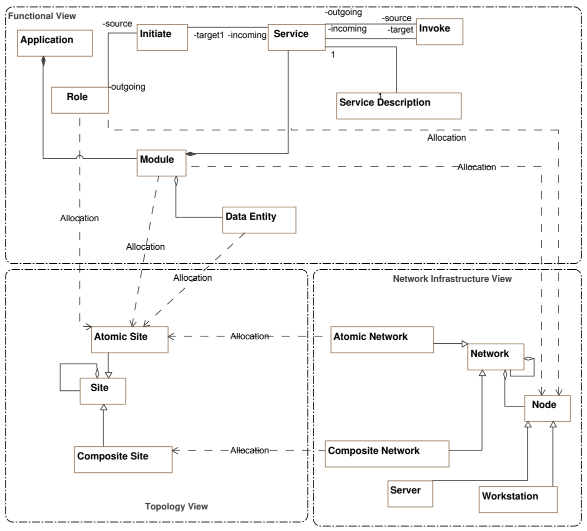


Figure 1. Functional, Topology and Network Infrastructure View Meta-model

For each service, a service description sub-view is defined indicating network infrastructure resources needed for its execution. The load imposed to network infrastructure resources each time the service is executed is expressed using requirements and is further described in NFR view.

Topology view (figure 1) facilitates allocation of software, data and people resources. It comprises of *sites*, organized in a hierarchical structure. Those belonging to the lowest level are characterized as atomic while others are characterized as composite. The allocation of modules, roles and data entities to sites corresponds to software architecture design. All of them are finally allocated to atomic sites. The load imposed to sites by the allocated resources is expressed using requirements and is further described in NFR view.

Network Infrastructure view represents the overall network decomposed to sub-networks. Servers, workstations and other network devices are associated with LANs at the lowest level of the hierarchy. Networks and network nodes are characterized by capacity properties, which should be specified by the system architect in order to satisfy all related requirements. Sites are allocated to networks. When an atomic site is allocated to an atomic network, functional view entities allocated in it must be specifically allocated to network nodes belonging in it.

3. Non-functional requirements classification

NFR view comprises non-functional requirements relevant to EIS architecture design. They are progressively defined during model-based EIS Architecture design tasks. Three main categories are supported: *performance*, *physical* and *specific quality* [7]. Performance requirements are emphasized, since they are essential in EIS architecture design.

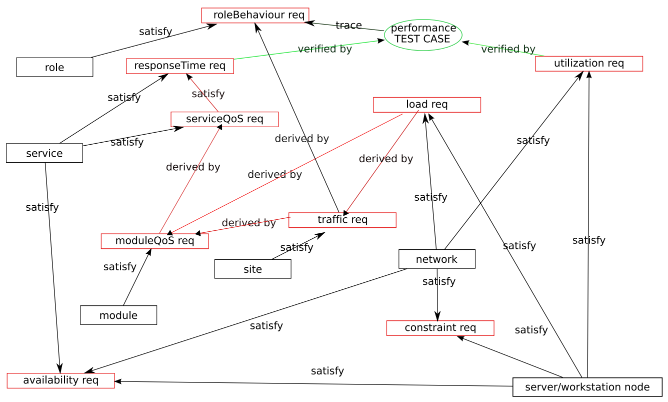


Figure 2. NFR View Meta-model

Performance requirements are further decomposed to *behavior*, *load* and *utilization*. Utilization requirements are associated with Network Infrastructure view and regard the proportion of network infrastructure resources used by applications during normal operation or extreme conditions. Behavior requirements deal with service behavior and are time-related (e.g. response times). They affect Functional view. Two of them are defined, namely *responseTime*, indicating the time interval within which a service should complete its execution, and *roleBehavior*, indicating activation patterns for roles defined within Functional view. Load requirements concern the load imposed to other EIS resources by EIS components allocated to them. Load requirements are defined in all views. Most of them are derived requirements, which are calculated using properties of other load requirements. Four different load requirements are defined, namely *serviceQoS*, *moduleQoS*, *traffic* and *load*.

Regarding physical requirements, indicating constraints imposed on design decisions by existing hardware resources, we focus on those concerning *capacity*. Capacity, indicating limitations of the hardware and their impact to the system, is related to Network Infrastructure view. Regarding specific quality requirements, we consider only *availability* requirements. They are associated with Network Infrastructure view, where availability deals with hardware aspects. Availability requirements may also be defined for applications within Functional View.

NFRs and the way they are interrelated to each other as well as to other entities belonging in Functional, Topology and Network Infrastructure views are depicted in figure 2. In the following, NFRs are analytically presented grouped by EIS Architecture view they are satisfied by.

Functional View Requirements: *RoleBehavior* requirement describes different user behavior, e.g. when, with what probability and how frequent a user initiates services. A role initiates services, while each service satisfies a *responseTime* requirement. The service requires EIS resources for its effective execution, expressed in terms of Quality of Service (QoS) it should receive from the underlying infrastructure. The serviceQoS requirement indicates the amount of processed, stored or transferred information a service requires during its execution. Consequently, the serviceQoS properties are average and maximum estimations of *traffic, processing* and *storage* QoS needed for the service execution. The QoS for each service is defined by the system architect, taking into account that it should satisfy corresponding *responseTime* requirement. ModuleQoS requirement describes the QoS needed for the module execution. It bears the same properties as serviceQoS and is apparently derived by the serviceQoS requirements belonging in the same module. Moreover, moduleQoS requirement properties are calculated as the aggregation of the values of the corresponding serviceQoS requirement properties.

Topology View Requirements: Sites satisfy traffic requirements, indicating the amount of information exchange between the allocated modules. Traffic requirement is described in terms of incoming, outgoing and exchanged traffic. Maximum and average values are computed. It is derived from moduleQoS and roleBehavior performance requirements as indicated in figure 2 and it is computed each time there is a change in allocations performed within Topology View.

Network Infrastructure View Requirements: Networks and network nodes are characterized by capacity indications, for example throughput, storage, speed or processing power. Their definition by the system architect must take into account constraints applied by existing infrastructure, availability, utilization and load requirements, as indicated in figure 2. Load requirements are computed based on ModuleQoS and Traffic requirement properties satisfied by entities allocated to the specific network infrastructure component (for example modules allocated to a specific network node).

In order to effectively define EIS Architecture, the system architect should ensure that all performance requirements are fulfilled. In SysML a test case is used to determine whether the system meets specifications placed by requirements. A test case is a set of conditions or variables which will be tested to ensure requirements are met. Thus, as indicated in figure 2, a *Performance Test Case* should be used to verify specific requirements, as *responceTime* or *Utilization* indicating the effective EIS operation. Since the verification of EIS Architecture design is performed using simulation, *Performance Test Case* is used for information exchange with the simulation environment.

4. SysML Profile Definition

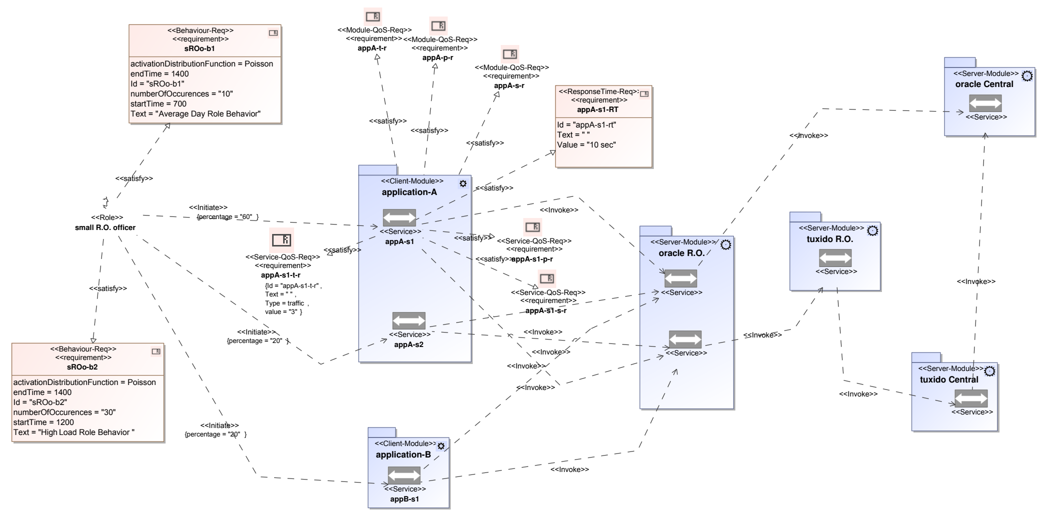
In the proposed SysML profile each view is depicted using a discrete diagram. Functional, Topology and Network Infrastructure views are described using hierarchical block-definition diagrams. SysML blocks can be used throughout all phases of system specification and design, and can be applied to many different kinds of systems. Entities belonging in these three views are related only using stereotypes of *SysML allocate* relationship, as indicated in 2. Requirements View is depicted using a Requirement diagram. Requirements are related to entities of all other views using *SysML satisfy* relationship. The profile is implemented using MagicDraw modeling tool [17].

NFR view is defined in SysML as a stereotype of a Requirement Diagram. Requirements in SysML are described in an abstract, qualitative manner, since they are defined using a name and a description. In the case of EIS Architecture Design non-functional requirements and especially performance requirements should be more accurately describe using quantitative properties. Furthermore, derived requirement properties should be automatically computed by combining specific attributes of requirement and allocation entities. Though, SysML provides for NFRs description, SysML requirement entity was heavily extended to effectively represent the quantitative aspects of performance requirements and the way they derive from each other.

5. Case Study

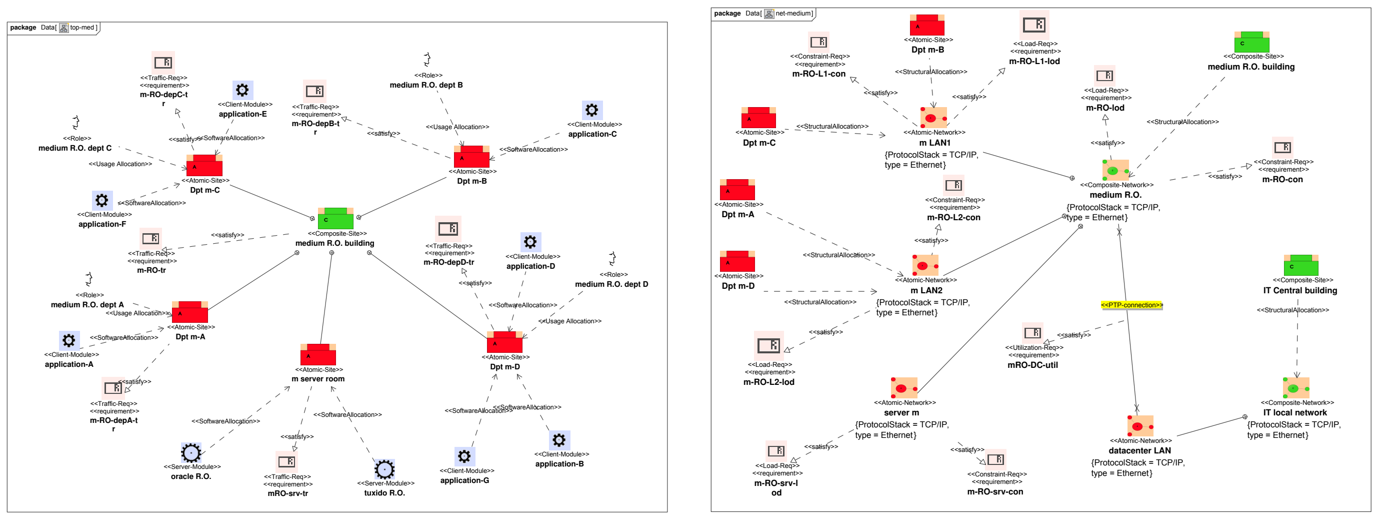
In the following we discuss the case of renovating a legacy information system supporting a large-scale public organization. The proposed model-based EIS Architecture design approach was applied to explore alternative architectures and their implications to the network infrastructure. One of the main objectives of system architecture re-design was to enhance application performance without rewriting the applications themselves. Since NFRs play a significal role in the re-design of the legacy information system, it was suggested to apply the proposed SysML profile, to explore related design desicions.

Functional View. In figure 3, a fraction of the Functional View is presented, where an officer working in a small regional office initiates three services belonging to two different modules (applications). As shown in the figure, this role satisfies two role behavior requirements. One of them refers to average day behavior and the other for a heavy load behavior. Using these requirements it is possible to test system performance under different conditions (workloads). **Topology view** depicts the structure of regional offices. A fraction of the Topology view corresponding to a medium office is depicted in figure 4. Traffic requirements satisfied by every site are computed and presented in the diagram. A server room is established in all regional offices. For each atomic site, all modules running in it are presented. The same is applied for the roles, which are presented as usage allocations. **Network Infrastructure View** represents existing network topology. Constraint requirements are used to depict existing infrastructure restrictions. Each regional office is connencted with IT datacenter through point-to-point (PTP) connections (figure 4). Every network is related



<<Behaviour-Req>> <<requirement>> sROo-b1
activationDistributionFunction = Poisson
endTime = 1400
Id = "sROo-b1"
numberOfOccurences = "10"
startTime = 700
Text = "Average Day Role Behavior"
<<Module-QoS-Req>> <<requirement>> appA-t-r
<<Module-QoS-Req>> <<requirement>> appA-p-r
<<Module-QoS-Req>> <<requirement>> appA-s-r
<<ResponseTime-Req>> <<requirement>> appA-s1-RT
Id = "appA-s1-rt"
Text = " "
Value = "10 sec"
<<Server-Module>> oracle Central
<<Service>>
<<Role>> small R.O. officer
<<Initiate>> {percentage = "60" }
<<Client-Module>> application-A
<<Service>> appA-s1
<<Service>> appA-s2
<<Service-QoS-Req>> <<requirement>> appA-s1-t-r
<<Service-QoS-Req>> <<requirement>> appA-s1-p-r
<<Service-QoS-Req>> <<requirement>> appA-s1-s-r
<<Server-Module>> oracle R.O.
<<Server-Module>> tuxido R.O.
<<Server-Module>> tuxido Central
<<Behaviour-Req>> <<requirement>> sROo-b2
activationDistributionFunction = Poisson
endTime = 1400
Id = "sROo-b2"
numberOfOccurences = "30"
startTime = 1200
Text = "High Load Role Behavior "
<<Initiate>> {percentage = "20" }
<<Initiate>> {percentage = "20" }
<<Client-Module>> application-B
<<Service>> appB-s1
<<satisfy>>
<<Invoke>>

Figure 3. Functional View



package Data[top-med]
<<Composite-Site>> medium R.O. building
<<Atomic-Site>> Dpt m-C
<<Atomic-Site>> Dpt m-B
<<Atomic-Site>> Dpt m-A
<<Atomic-Site>> Dpt m-D
<<Atomic-Site>> m server room
<<Client-Module>> application-E
<<Client-Module>> application-C
<<Client-Module>> application-F
<<Client-Module>> application-D
<<Client-Module>> application-A
<<Client-Module>> application-G
<<Client-Module>> application-B
<<Server-Module>> oracle R.O.
<<Server-Module>> tuxido R.O.
<<Role>> medium R.O. dept B
<<Role>> medium R.O. dept C
<<Role>> medium R.O. dept A
<<Role>> medium R.O. dept D
<<Traffic-Req>> <<requirement>> m-RO-depC-tr
<<Traffic-Req>> <<requirement>> m-RO-depB-tr
<<Traffic-Req>> <<requirement>> m-RO-tr
<<Traffic-Req>> <<requirement>> m-RO-depD-tr
<<Traffic-Req>> <<requirement>> m-RO-depA-tr
<<Traffic-Req>> <<requirement>> mRO-srv-tr
<<Usage Allocation>>
<<SoftwareAllocation>>
<<satisfy>>
package Data[net-medium]
<<Atomic-Site>> Dpt m-B
<<Atomic-Site>> Dpt m-C
<<Atomic-Site>> Dpt m-A
<<Atomic-Site>> Dpt m-D
<<Constraint-Req>> <<requirement>> m-RO-L1-con
<<Load-Req>> <<requirement>> m-RO-L1-lod
<<Atomic-Network>> m LAN1 {ProtocolStack = TCP/IP, type = Ethernet}
<<Composite-Site>> medium R.O. building
<<Load-Req>> <<requirement>> m-RO-lod
<<Composite-Network>> medium R.O. {ProtocolStack = TCP/IP, type = Ethernet}
<<Constraint-Req>> <<requirement>> m-RO-con
<<Constraint-Req>> <<requirement>> m-RO-L2-con
<<Atomic-Network>> m LAN2 {ProtocolStack = TCP/IP, type = Ethernet}
<<PTP-connection>>
<<Composite-Site>> IT Central building
<<Load-Req>> <<requirement>> m-RO-L2-lod
<<Utilization-Req>> <<requirement>> mRO-DC-util
<<Atomic-Network>> server m {ProtocolStack = TCP/IP, type = Ethernet}
<<Atomic-Network>> datacenter LAN {ProtocolStack = TCP/IP, type = Ethernet}
<<Composite-Network>> IT local network {ProtocolStack = TCP/IP, type = Ethernet}
<<Load-Req>> <<requirement>> m-RO-srv-lod
<<Constraint-Req>> <<requirement>> m-RO-srv-con
<<StructuralAllocation>>
<<satisfy>>

Figure 4. Topology & Network Infrastructure View

to a constraint requirement, which describes the existing speed of the network and a load requirement, which describes "how much traffic" do the applications that belong to that network require. Moreover, a utilization requirement is satisfied by the point-to-pont connection between regional offices and the datacenter. Eventually, the designer in order to define the connection speed between two networks, load requirements of the networks and utilization of the network connection must be taken into account. Load requirements depend on the server distribution, meaning that if the servers are distributed across the local offices, the load will be higher whereas if the applications are web-based, load will be less. For a network of a higher hierarchy, load requirement is derived by the load requirements of the lower lever networks. In order to calculate PTP connection utilization two parameters have to be defined: the network load and the network connection speed. For a network, a corresponding load requirement is assigned which is derived by the traffic requirements for each site allocated to that network.

The network architecture is predefined. The system architect was enabled to explore two different database architecture scenarios. The first one was to eliminate local database servers and consolidate them in the IT Center, without intervening with database architecture. That is, a local database is kept for every regional office, but all of them are hosted in the IT Center where the central database is also maintained, while local Transaction Monitoring Servers still operate in Regional offices. The scenario has no effect on the manner applications operate. Though it resulted in considerably increasing network traffic over the

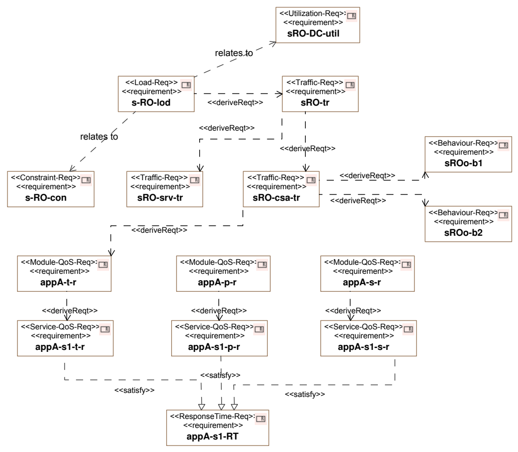


Figure 5. NFR View - Small Regional Office Example

point-to-point connections between regional offices and the IT Center, while the processing power needed to support all databases is consolidated in the IT Center. The second one was to eliminate local databases and establish one central database. In this case local Transaction Monitoring Servers still operate in regional offices, but the applications need minor rewriting. Though both scenarios have the same affect to the network architecture, the second one results in a more efficient architecture solution for the Datacenter since there is no need to synchronize databases hosted on the same servers, which consequently affects application performance. Since it was estimated that performance improved almost by 1/3 by the second scenario, it was decided to apply it despite the minor application rewriting involved.

NFR View integrates all NFRs from all views and their relations. These relations are in accordance with the general requirements relations as depicted in figure 2. Different views are interrelated through the relations of their requirements, which is accomplished by the NFR diagram. Figure 5 presents an example of the requirements of a small regional office and their relations.

6. Conclusions & Future Work

In practice, EIS Architecture design is usually performed by properly integrating EIS components already defined by other stakeholders in an efficient manner. Thus, NFRs must be emphasized. The provision of a well-defined model to represent NFRs and the relations between them in a separate view, enhances the system designer's perception of the way specific design decisions may affect others, as they are practically represented through derived requirements.

Though, SysML provides for requirements description, this is supported in a abstract fashion. During system design, NFRs and particularly performance requirements have to be extensively described using quantitative parameters, while the way they are derived from each other should be expressed in a quantitative manner. To provide such a functionality, SysML requirement entity used heavily extended.

The proposed profile is currently tested in other case studies as well. Furthermore, we are exploring the proposed solution validation using simulation in an automated fashion. Information exchange between the simulation environment and SysML profile is explored.

References

[1] INCOSE. INCOSE Handbook SE Process Model, September 2003. http://g2sebok.incose.org/.

[2] A. Aurum and C. Wohlin. *Engineering and Managing Software Requirements*. Springer, 2005.

[3] E. R. Byrne. IEEE Standard 830: Recommended Practice for Software Requirements Specifications, 1998.

[4] A. v. Lamsweerde. Goal-Oriented Requirements Engineering: A Guided Tour. In *Fifth IEEE International Symposium on Requirements Engineering (RE'01)*, p 249, 2001.

[5] J. Mylopoulos, L. Chung and B. Nixon. Representing and using nonfunctional requirements: A process-oriented approach. In *IEEE Trans. Softw. Eng*, 18(6):483-497, 1992

[6] L. Zhu and I. Gorton. Uml profiles for design decisions and non-functional requirements. In *SHARK-ADI '07*, p 8, 2007. IEEE Computer Society.

[7] M. Glinz. On non-functional Requirements. 15th IEEE International Requirements Engineering Conference, 2007.

[8] C.-W. Ho, L. Williams, and B. Robinson. Examining the relationships between performance requirements and "not a problem" defect reports. In *RE '08*, p 135–144, 2008. IEEE Computer Society.

[9] J. A. Estefan. *Survey of Model-based Systems Engineering (MBSE) Methodologies*. INCOSE MBSE Focus Group, May 2007.

[10] M. Nikolaidou, A. Tsadimas, N. Alexopoulou, D. Anagnostopoulos. Employing Zachman Enterprise Architecture Framework to Systematically Perform Model-Based System Engineering Activities In *HICSS-42* ,p 1–10, 2009.

[11] O. M. G. Inc. Systems Modeling Language (SYSML) Specification. Version 1.0, September 2007.

[12] S. Leist and G. Zellner. Evaluation of current architecture frameworks. In H. Haddad, ed, *SAC*, p 1546–1553. ACM, 2006.

[13] M. Nikolaidou and N. Alexopoulou. Enterprise Information System Engineering: A Model-Based Approach Based on the Zachman Framework. In *HICSS'08*. IEEE Computer Society, 2008.

[14] A. Fatolahi and F. Shams. An investigation into applying UML to the Zachman Framework. *Information Systems Frontiers*, 8(2):133–143, 2006.

[15] C. M. Pereira and P. Sousa. A method to define an Enterprise Architecture using the Zachman Framework. In H. Haddad, A. Omicini, R. L. Wainwright, and L. M. Liebrock, editors, *SAC*, p 1366–1371. ACM, 2004.

[16] OMG. UML Profile for Modeling, Quality of Service and Fault Tolerance Characteristics and Mechanisms, September 16, 2004, omg.org

[17] MagicDraw UML. http://www.magicdraw.com/.

[18] I. Ozkaya. Representing requirement relationships. In *REV '06*, p 3, 2006. IEEE Computer Society.