A Consistent Framework for Enterprise Information System Engineering

M. Nikolaidou[1], N. Alexopoulou[12], A. Tsadimas[1], A. Dais[1], D. Anagnostopoulos[1]
{mara@hua.gr, nancy@hua.gr, tsadimas@hua.gr,adais@hua.gr, dimosthe@hua.gr}

[1] *Harokopio University of Athens,*
El. Venizelou Str, 17671Athens, Greece

[2] *Department of Informatics and Telecommunications,*
University of Athens, Panepistimiopolis, 15771, Athens, Greece

Abstract

System engineering is the process of defining the desired architecture of a system and exploring performance requirements, ensuring that all system components are identified and properly allocated and system resources can provide the desired performance. A consistent framework for enterprise information engineering, compatible to Zachman framework is proposed. It consists of a metamodel describing different system views and the relations between them, a corresponding methodology of discrete stages, performed by the system designer or software tools, and a UML 2.0 profile for view representation.

1. Introduction

The Zachman framework for Enterprise Architecture provides a taxonomy for relating the concepts that describe the Enterprise to the concepts that describe the Information System supporting it and its implementation [1]. It can be used as a guide for establishing an Enterprise Information System (EIS) to ensure that Enterprise requirements are met. The framework identifies the stakeholders involved in this effort and provides corresponding viewpoints according to the stakeholder perspective. For each viewpoint a system model is defined consisting of different views related to different aspects (e.g. data, function, network, etc). The framework identifies the scope of each view and the main entities participating in it. It does not provide a typical model for the definition of each view, neither identifies a representation language, thus it is technology neutral. It is also simple and comprehensive. Although it is very analytical supporting 30 different views, the definition of such a large number of views and their relations might be confusing.

System engineering is the process of defining the desired architecture of a system and exploring performance requirements, ensuring that all system components are identified and properly allocated and system resources can provide the desired performance. Enterprise information system engineering is a task accomplished by the system designer, thus related issues should be explored within the *System Model* of the Zachman framework corresponding to the designer viewpoint, although system model is not limited in system engineering issues. For all the models supported, Zachman framework proposes six (6) alternative views.

In this paper, we present our effort to establish a consisted framework for enterprise information engineering that is compatible to Zachman framework and especially the System Model. The framework provides:

- A metamodel describing different views. The suggested views correspond to Zachman System Model views. The relations between them are strictly defined using constraints
- A methodology for EIS engineering based on the proposed views. The methodology takes advantage of the relations identified between views.
- The integration of software tools performing specific configuration tasks.
- A representation meta-model for all the views defined, which facilitated an integrated, easy-to-use interface for the system designer.

2. EIS Architecture Frameworks

The widely referenced Enterprise Architecture framework of Zachman [2] specifies the establishment of information systems starting from the identification of the enterprise's business objectives. System engineering issues are addressed in the System Model raw of the Zachman's matrix. It should be noted, that system designer may actually work concurrently with the system developer (the builder of the model), although in general system design should be performed prior its implementation. In many cases, during system design, although system architecture

This research was supported by Pythagoras program (MIS 89198) co-funded by the Greek Government (25%) and the European Union (75%).

is defined and the services provided by the distributed applications are identified, detailed software design and implementation is considered in the builder model. System engineering issues should be dealt with independently of the status of software development process.

Rational Unified Process – System Engineering (RUP SE) [3] is dealing with issues related to Zachman's System Model. For each viewpoint four (4) different views are constructed. These views although are independent, should be related at least by the refinement relation [4], in order to ensure consistency between the different detail levels of the System Model. UML 1.3 diagrams are employed for the illustration of these views. For the representation of certain concepts not directly supported in UML 1.3, such as system worker and locality, which denotes a grouping of physical resources providing logical services, the RUP SE framework defines appropriate stereotypes. However, RUP SE does not provide a formal metamodel or UML profile for view representation.

Furthermore, the plethora of views all referring to the same system model, although providing the capability of detail system description, is complex to manage. The most important issue is that as they refer to the same system model, they should be kept aligned and consistent with respect to each other. In order to ensure consistency and avoid the loss of information critical during system design, various types of relations between different views (and corresponding models) should be enforced (e.g. equivalence or refinement relations). We propose the definition of a smaller number of views, formally related to ensure consistency between them. Furthermore, we propose to avoid further decomposing the System Model into sub-models to reduce complexity. The discrete stages of System Engineering process (in correspondence to RUP SE successive phases, namely *Context*, *Analysis*, *Design* and *Implementation*) can identified and coordinated using constraints embedded in the metamodel describing the views corresponding to the system. [5].

The framework also provides:

- A metamodel describing different views and the relations between them (EIS metamodel). These relations are strictly defined using constraints.
- A methodology for EIS engineering based on the proposed views. The methodology consists of discrete stages performed by the system designer, software tools or a combination of both. Taking advantage of the formal definition of relations identified between views, system engineering stages may by invoked by metamodel constraints, ensuring that each of them can be independently performed.
- A UML representation for all defined views. A UML 2.0 profile is defined for this purpose (EIS engineering profile)

3. EIS Engineering Framework

The framework is based in three complementary views:

Functional View is used to describe functional specifications (e.g. system architecture, user behavior and application requirements). System architecture refers to the architectural model adopted. In the case of EIS, multi-tiered client-server models are described. Services provided by each application tier (called modules) are also defined. User behavior is modeled through user profiles defining the behavior of different user groups and their performance requirements. Application requirements are described in terms of quality of service (QoS) requirements imposed to the network infrastructure, e.g. amount of data processed, transferred or stored. Each service is described in a greater level of detail through the *service description* sub-view.

Topology View facilitates the definition of system access points and the resource allocation and replication. The term *site* is used to characterize any location (i.e. a building, an office, etc.). As such, a site is a composite entity which can be further analyzed into subsites, forming thus a hierarchical structure. Functional and topology views are related. Resources (e.g. processes and files) correspond to services and data described through functional view and are located into sites.

Physical View refers to the aggregate network. Network *nodes* are either *workstations* allocated to users or *server* stations running server processes. Topology and Physical views are interrelated. Both are decomposed to the same hierarchical levels of detail. At the lowest level, network nodes are related to processes/data replicas.

Both views can be either defined by the system designer or automatically composed by configuration tools. The introduction of progressive site refinement for both sites and networks, corresponding to Topology and Physical View respectively, as well as the mapping of site range onto network range, enables the identification of dependencies between them. These dependencies must be formally defined.

EIS engineering framework facilitates the following discrete stages of System Engineering process:

1. System requirement definition.
2. Resource (process/data) allocation and replication policy definition.
3. Network architecture design.
4. Performance evaluation of the proposed solution (prior to implementation). Although it is not a necessity, it is certainly useful.

As resource allocation and network design problems cannot be independently solved, stages (2) and (3) are repeatedly invoked for different abstraction levels until an acceptable solution is reached [5]. Both resource allocation and network architecture problems are usually supported by automated or semi-automated tools using mathematics, heuristics or a combination of both. These tools may be repeatedly invoked for different abstraction levels [6, 7]. The system designer may perform or partially perform these tasks on his own, thus both options must be supported. To evaluate system performance, a simulation tool as the one described in [8] can be used. The simulator uses as input the overall system model and produces performance results. Since each of these tools supports its own representation metamodel (for example queuing networks, Petri-nets, objects), there is a need to properly create and instantiate the "internal" system model prior invoking the tool. The proposed methodology stages along with EIS model consisting of the predefined views are presented in figure 1.

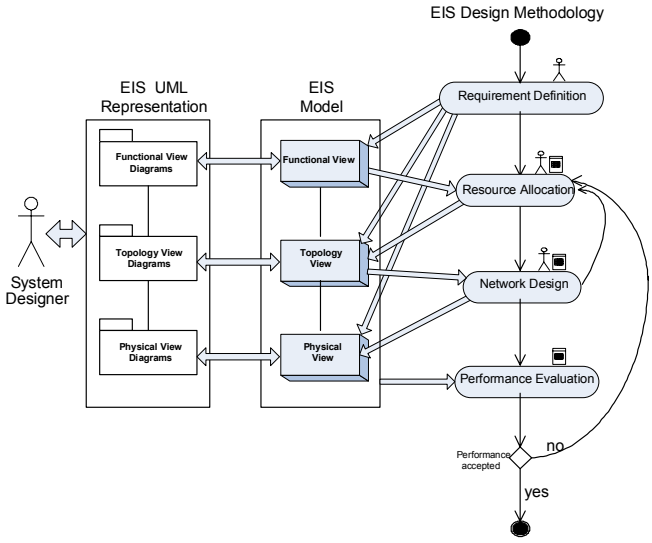

Figure 1: EIS Engineering Framework

Discrete stages receive/modify information from/to specific system views, as depicted by the arrows between them. The relation between views and between stages is also depicted in the figure. Requirement definition is the initial stage and corresponds to the definition of system architecture and application requirements (functional view), the system access points (topology view) and existing network architecture – if any- (physical view). A metamodel is provided for the formal definition of views and the relations between them. Each view is represented by one or more UML diagrams properly extended, thus a corresponding UML 2.0 profile is defined. Relations between views must also be described in the UML profile.

Figure 2 depicts the mapping between the proposed framework and Zachman's focal points of the System Model viewpoint in terms of their views.

Zachman Framework - System Model

People	Function	Data	Process	Network	
Functional			*not addressed yet*	Topology	Physical

EIS engineering framework

Figure 2: Mapping to Zachman Framework

3.1 EIS Meta-model

For each distributed application operating in the EIS, a discrete *Functional View* is defined. Applications are conceived as sets of interacting *modules*. Each module offers specific *services*. *Data entities* are defined to indicate portions of data used by application modules. User behavior is also described in the Functional View, through *user profiles*. For each module service, the requirements imposed to the network infrastructure must be defined. Thus the portion of data processed, stored or transferred must be estimated. Also other services participating in its implementation must be identified. This is performed using a set of predefined *operations*, sketching service functionality and describing its needs for *processing, storing* and *transferring* (called elementary operations). Since it is difficult for the system designer to estimate the elementary operations describing service requirements, an operation library, named *Operation Dictionary* is provided.

Physical view comprises the network infrastructure. The overall *network* is decomposed to subnetworks producing thus a hierarchical structure. LANs typically form the lowest level of the decomposition. Nodes, such as servers and workstations are associated with LANs of the lowest level. Nodes may include a *processing unit* and a *storage unit*.

Topology view comprises sites, processes (defined as instances of application modules) and users (defined as instances of user profiles). Two types of sites are supported: composite, composed by others, and atomic, not further decomposed, constituting therefore the lowest level of site hierarchy. Users, processes and files are associated to atomic sites. The site hierarchy should correspond to the network hierarchy depicted in the physical view, while processes, files and users are related to nodes included in the physical view.

The metamodel itself contains relationships and restrictions inflicted between system entities belonging to the same or different views, which may lead to a specific stage invocation. Embedding restrictions within the metamodel facilitates EIS engineering process management taking into account the overall system model and not the specific system view corresponding to a

discrete stage. Thus, the overall process becomes more effective, since discrete stage (and corresponding tool) dependencies are depicted within the model as view dependencies and consequently they are easily identified.

3.2 EIS UML 2.0 Profile

In order to provide a standard method to represent system views and facilitate the designer to interact with them, a UML 2.0 profile [9, 10] was defined. UML 2.0 diagrams are used to represent different aspects of views. EIS entities are depicted as UML model elements included in the corresponding diagram. They may be created by the system designer through the UML modeling tool or automatically by software tools. UML 2.0 stereotypes are used to represent EIS entities, properly defined to include additional properties and constraints. Essentially, the concepts of the metamodel are reflected onto the stereotype attributes and constraints. Attributes convey the information required to describe EIS metamodel entities (e.g. *throughput, activationFrequency, processingPower* etc.). Constraints, which are extensively used within the profile, represent relationships and restrictions between metamodel entities maintaining model consistency. Constraints mainly facilitate:

1) Automatic computation of specific attribute values
2) Limiting attribute value range
3) Relating attribute values of specific elements to attribute values of other entities belonging to the same or other UML diagrams (implementing thus the linkage between different models) and
4) Model validation in view and overall model level.

Attributes and constraints for each stereotype are analytically introduced in [11]. Functional View is represented through UML component diagram, since component diagrams are eligible for depicting system functionality at a logical level. Concerning service description sub-view, it is represented through an activity diagram, as it involves flow of operations. UML communication diagrams, which depict interaction between entities, are suitable for the representation of Operation Dictionary, since the latter involves interactions between operations showing in particular invocation order and parameter passing between them. Physical View comprises the network infrastructure. As such it is depicted through UML deployment diagrams, which are commonly used to represent network architectures [12]. Lastly, the representation of Topology View is based on UML component diagrams.

4. Conclusions

A consistent framework for EIS engineering was proposed. It consists of a metamodel describing proposed system views and the relations between them, a corresponding methodology consisting of discrete stages performed by the system designer or software tools and a UML 2.0 profile for view representation. Constraints impose restrictions and relationships between entities participating in different views, facilitating a formal mapping between them.

5. References

[1] Zachman A. J., "A Framework for Information Systems Architecture" *IBM Systems Journal*, Vol. 31, No. 3, pp.445 –470, 1999.

[2] Sowa F. J. and Zachman A. J., "Extending and formalizing the Framework for Information Systems Architecture" *IBM Systems Journal*, Vol. 38, No. 2&3, pp.590 – 616, 1992.

[3] Murray Cantor, Rational Unified Process for Systems Engineering – Part II: System Architecture, 2003.

[4] Dijkman R.M., Quartel D.A.C., Pires L.F., Sinderen M.J., "An Approach to Relate Viewpoints and Modeling Languages", in *Proceedings of the 7th International Enterprise Object Computing Conference (EDOC'03),* IEEE Computer Press, 2003.

[5] Nikolaidou M., D. Anagnostopoulos, "A Systematic Approach for Configuring Web-Based Information Systems", *Distributed and Parallel Database Journal*, Vol 17, pp 267-290, Springer Science, 2005.

[6] Graupner S., Kotov V., Trinks H., "A Framework for Analyzing and Organizing Complex Systems", in *Proceedings of the 7th International Conference on Engineering Complex Computer Systems (ICECCS'01)*, IEEE Computer Press, 2001.

[7] Nezlek G.S., Hemant K.J., Nazareth D.L., "An Integrated Approach to Enterprise Computing Architectures", *Communications of the ACM*, Vol 42, No 11, ACM Press, 1999.

[8] Nikolaidou M. Anagnostopoulos D., "A Distributed System Simulation Modeling Approach", *Simulation Practice and Theory Journal*, Vol. 11, No 4, Elsevier Press, 2003.

[9] OMG Inc, UML Superstructure Specification, 8/10/2004.

[10] OMG Inc, UML 2.0 Infrastructure Specification, 30/4/2004.

[11] Alexopoulou N., Nikolaidou M, et al, "Introducing a UML Profile for Distributed System configuration", *in Proceedings of the 8th International Conference on Enterprise Information Systems (ICEIS'2006),* 2006.

[12] Kaehkipuro P., "UML-Based Performance Modelling Framework for Component-Based Distributed Systems", Lecture Notes in Computer Science 2047, *Performance Engineering*, Springer-Verlag, 2001.

.