Enterprise Information Systems Configuration: Emphasizing the Symbiotic Relationship between Applications and the Underlying Network

Mara Nikolaidou
University of Athens
Panepistimiopolis, 15784 70
Athens, Greece
Email: mara@di.uoa.gr

Dimosthenis Anagnostopoulos
Harokopion University of Athens
El. Venizelou Str, 17671
Athens, Greece
Email: dimosthe@hua.gr

Abstract

Enterprise information systems consist of interrelated Intranet-based and Internet-based applications, thus the Web platform serves well as the middleware for their development. Even though, in many cases, discrete applications operate efficiently, the overall system performance is less than expected. A potential cause is that, configuration issues although interrelated are solved in isolation. The architecture and performance characteristics of the underlying network affect application configuration. Thus, the symbiotic relationship between application and network architecture must also be explored. We argue that, in order to effectively configuring and evaluating enterprise information systems, network infrastructure restrictions must be incorporated within application configuration process and vise versa. An effective four-stage methodology is proposed for this purpose. It is important to note the significance of a consistent model for the representation of system entities throughout all stages. UML-like notation is used for system representation. Alternative views of the system emphasizing specific configuration stages are offered through UML package representation. Since the modeling scheme is extendable, the adaptation of UML constructs simplifies the process of extending or customizing the model. A case study where the proposed methodology is used for the configuration a large-scale medical information system and the experience obtained are also presented.

1. Introduction

Enterprise information systems (EISs) consist of a combination of interrelated Intranet-based and Internet-based applications, built on multi-tiered client-server models [1]. Some of the main characteristics of EISs are: a) their wide scale, as they operate upon a variety of network platforms, b) their complexity, as they consist of cooperating heterogeneous distributed applications (e.g database applications, workflow systems, web services) and c) their extendibility, as they expand gradually to satisfy evolving user requirements. Security issues should also be considered. End-users in an enterprise environment have to interact with a multiple applications, thus, it is important to provide a common "look and feel" to application interfaces. The Web platform serves well as a common access environment for all the applications operating within enterprise Intranet boundaries, while, at the time, acts as the middleware integrating Intranet-based and Internet-based applications.

Significant vendors, such as Oracle and IBM, provide web-based software development platforms, such as Oracle Application Server [2] and IBM WebSphere [3], which facilitate application integration and enable users to access them through a common interface using a web client. In such cases, the first and second application tiers are implemented using the WWW platform [1]. The first tier, e.g. the web client, is only responsible for user interaction, while the second tier, e.g. the web server, is responsible for invoking the proper application service, obtain results and forward them to the user as HTML/XML pages or fields. Other tiers, providing specific application functionality, are implemented as sets of cooperating services distributed on different servers and are based on a variety of architectures discussed in [4].

Even though vendors actively promote information system development using the aforementioned software platforms, the proposed solutions, although expensive, often do not provide the desired performance [5]. A potential cause is that, as most enterprise information systems expand gradually, system extensions are performed without ensuring the overall system performance. Furthermore, distributed applications,

although accessed by a common web interface, are characterized by their internal complexity, the impact of which cannot be determined using trivial mathematics. To ensure their performance, their configuration dependencies must by identified and explored before employing specific application or data replication and synchronization policies. The architecture and performance characteristics of the underlying Intranet platform and Internet connections also affect application configuration. Thus, the symbiotic relationship between application and network architecture must also be explored. We, thus, argue that, in order to effectively configuring and evaluating enterprise information systems, network infrastructure restrictions must be incorporated within application configuration process and vise versa.

A systematic approach for the configuration of any complex information system offers considerable capabilities providing decision making support to the system designer to ensure system efficiency when building a new system or extending an existing one [6, 7, 8]. In the following, we identify the discrete stages of enterprise information system configuration procedure and their dependencies and discuss alternative methods and techniques to automate each of them. Each stage addresses a specific issue, such as application functionality description, resource allocation and replication, network topology design and performance evaluation. Each of these issues is usually handled in isolation, resulting in poor performance. To integrate handling of all configuration problems, it is of extreme significance to use a consistent model for the representation of system entities throughout all stages. We propose such a model, which enables the identification of unclear application specific dependencies between discrete stages, since it is used as the reference framework to estimate application requirements, apply resource allocation and replication policies and construct network topology. This model must enable the description of any kind of application, thus be extendable, while it should also be easy to realize in various software tools used to automate discrete configuration stages. It should also be easy to use by the designer providing system functionality specifications.

Since system designers are usually familiar with UML [9], as a process and data modeling language, it was decided to use UML-like notation to model all aspects of enterprise information configuration process in a multi-layer fashion by integrating different diagram types [10]. In [11, 12], UML sequence diagrams facilitate the description of client-server architectures emphasizing the triggering of processes and the information exchange between them. However, the description of internal process functionality is not facilitated. Furthermore, user behavior should also be incorporated in the model. Finally, the dependencies between applications and network infrastructure must be modeled. Thus, different system views must be provided facilitating their identification.

The rest of the paper is organized as follows: In section 2, we briefly discuss information system configuration process and our approach to automate it. Configuration stages are described and basic properties of the common model used to represent system entities are identified. In section 3, the UML-like modeling approach is introduced and the benefits obtained during system configuration and performance evaluation are presented. In section 4, we introduce the *site* concept used to indicate system access points and group application tiers. Sites are composite entities that may be constantly refined. Sites facilitate incorporating network infrastructure restrictions in application configuration, and visa versa. In section 5, a case study where the proposed model is used during the configuration of a large-scale medical information system is presented, while conclusions reside in section 6.

2. EIS Configuration Process

System configuration stages and their interaction are depicted in figure 1. *Functional configuration* (stage 1) corresponds to the description of system specifications. *Logical and physical configuration* (stages 2 and 3) deal with process/data allocation and replication policies and network topology design respectively. Resource allocation and network configuration problems cannot be independently solved. Thus, stages (2) and (3) are invoked iteratively until an acceptable solution is reached. In order to automate or semi-automate configuration stages, it is important to provide a consistent model for the representation of EIS entities throughout all configuration stages. The EIS model consists of a. *functional specifications* (e.g. application logic and user behavior), b. *physical specifications* (the underlying network) and c. their dependencies. The EIS model is progressively constructed as follows:

1. Functional specification and parts of the physical specification (i.e. referring to the existing pieces of computer/network infrastructure) are defined during functional configuration stage.
2. Logical configuration defines the dependencies between functional and physical specifications, as resource allocation and replication policies result in the allocation of processes and data instances to hardware components.

3. Physical configuration results in the creation of physical specifications.

We decided to adapt UML-like notation for EIS model, since UML a) is a widely accepted standard and most system designers are familiar with it, b) allows the graphical representation of specifications and c) facilitates the automated implementation of model extensions. Functional configuration is strongly related with model definition. Logical and physical configurations can be semi-automated using heuristics by appropriate decision-support software, for example IDIS [13] that facilitates the representation and exploration of resource allocation and network topology design algorithms combining mathematics and rules of thumb. System configuration stage must facilitate the *performance evaluation* (stage 4) of the proposed solution prior to implementation. If system performance specifications are not satisfied, logical and physical configuration stages must be repeatedly performed. To evaluate distributed system performance, the discrete event simulation tool described in [14] can be used. The *Information System Configuration Guide* is a prototype environment written in Java, which facilitates the management of the EIS model, the co-ordination of discrete configuration stages and the automation of functional configuration. It also contains a set of wrappers for properly initializing external software modules and facilitating data exchange using object-oriented representation.

Each EIS is modeled as an aggregation of interacting components, either primitive or composite, usually customized to depict the functionality of specific system components, based on the following assumptions:

- Distributed applications are built based on client-server models and consist of multiple tiers. The first and second tiers (e.g. web client and server) are implemented using Web technology. This reflects on process allocation policies, since a substantial part of application tiers must be close to the user. To achieve the required application performance, the *principle of locality* (i.e. keeping servers and data as close as possible to user) is widely applied. Replication techniques are employed to increase performance and availability, especially over the Internet. Replica synchronization is usually performed using asynchronous policies.
- The underlying network consists of heterogeneous Intranets and Internet connections integrated through TCP/IP protocol stack. Users have their own workstation (diskless or not). Server processes are executed on dedicated server nodes. Application performance is greatly influenced by individual server machine performance. The communication between user-related tiers (web tiers) is based on HTTP protocol.

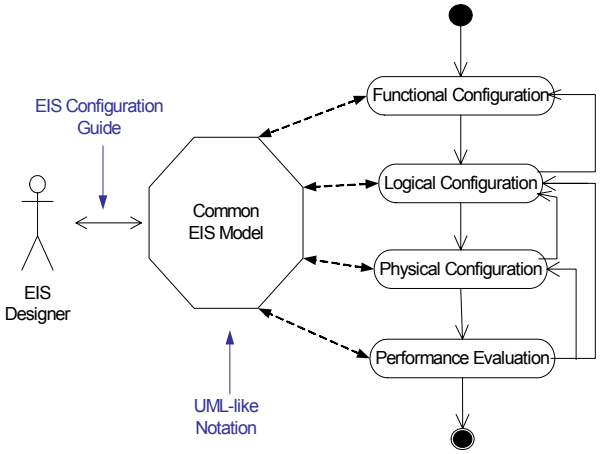

Figure 1: EIS Configuration Process

3. UML-like EIS Model

EIS architecture is modeled using three alternative views, the *application* and the *site view*, emphasizing functional specifications, and the *physical view*, emphasizing physical specifications. Different UML diagrams are used to represent each view. Adopting UML-like notation, each EIS entity is depicted by extending the properties of corresponding UML diagram entities using the *stereotype* mechanism. The overall UML class diagram corresponding to EIS model is presented in figure 2. It identifies a basic set of object types to describe functional and physical specifications and their relations. Gray rectangles represent first-level entities, corresponding to alternative system views. Further object types may be added by the designer to describe additional functionality by extending or restricting existing object behavior. Model extension is essential to enrich the model capability to describe custom applications and is performed through *Model Extension View*.

3.1. Physical View

It refers to the aggregate network. As indicated in figure 2, it consists of a *Network* entity represented as UML package. Network is a composite entity, which is repeatedly refined to represent network topology. Each *network* either consists of multiple *networks* and *internetworks* (1:N), or represents a simple Intranet LAN. Each internetwork represents an Intranet LAN, Intranet WAN or Internet connection. Network *nodes* are either *workstations* allocated to users or *server* stations, running server processes. Networks and inte The architecture and

performance characteristics of the underlying Intranet platform and Internet connections also affect application configuration. Thus, the symbiotic relationship between application and network architecture must also be explored. We, thus, argue that, in order to effectively configuring and evaluating enterprise information systems, network infrastructure restrictions must be incorporated within application configuration process and vise versa. rnetworks also include multiple *relay nodes* (1:N) depicting routing/switching functionality and one *channel* element, representing the communication link. Processing and relay nodes consist of individual *elements* corresponding to the three *elementary operations* supported in a network environment: *processing*, *storing* and *transferring* data. Specifically, processing nodes consist of one *processing*, one *storage* and one *communication* element, while *relay nodes* consist of a *processing* and multiple *communication elements* (1:N), one for each network they relay.

UML deployment diagrams are commonly used to represent network architectures [12]. In EIS model, networks and internetworks are represented through UML packages containing the corresponding deployment diagram. *UML arcs* represent channels. Network/internetwork nodes entities are represented as stereotypes of the *UML Node* entity by extending its semantics, e.g. the number of identical processing/relay nodes is included in the node entity of deployment diagrams. Network nodes are composite objects. Node elements are also represented as stereotypes of *UML Node* entity (figure 2).

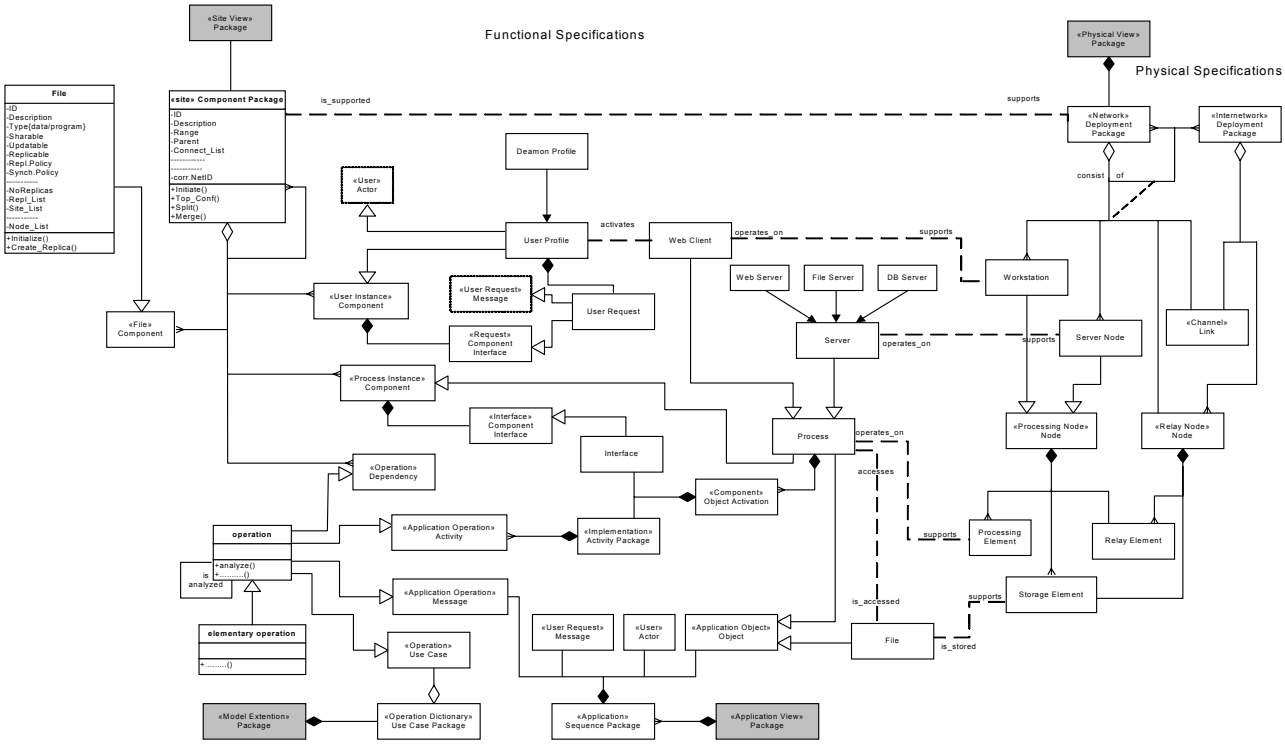

Figure 2: EIS model class diagram

3.2 Application View

Application View consists of the supported applications represented through UML packages containing the corresponding sequence diagram (figure 2). Applications are conceived as sets of interacting *processes* and the data repositories (i.e. *files)* accessed by them. The services composing the aggregate functionality of an application can be described using the *process* and *component* concepts. As a process may be activated in different ways (based on its input arguments), a *component* represents the specific set of tasks (or operations) executed when a process is activated in a certain way. In the proposed model, a *process* is thus composed by the *components* corresponding to all alternative activation ways. Processes are represented as stereotypes of *UML objects* and components as *object activations*. An example of the application view is

depicted in figure 3, where a simple database search is initiated by a web user through the proper CGI in the Web Server. Components are composed of two distinct parts:

- an *interface*, depicting the process activation mechanism
- an *implementation* part, comprising the tasks that occur upon process activation.

Component implementation is described using operations from a predefined *operation* set, that is, the *operation dictionary*. Component implementation is described using a *UML activity* diagram supporting both sequential and concurrent operation execution, where operations are represented as stereotypes of *UML activity* entities. In figure 3, the UML activity diagram for the *Simple Search* component of *Web Client* process is depicted. When double-clicking in the Web Client implementation in Application View (represented as a object activation in the corresponding sequence diagram), the system designer opens a new pop-up window containing the activity diagram corresponding to Web Client implementation part. The system designer may add a new operation in the diagram and define its properties, e.g. operation type and operation parameters, using a scroll menu. *UML message* entities are automatically added in the sequence diagram, representing Application View, between object activations to represent process interaction. Messages are labeled using the name of the operation initiating process activation.

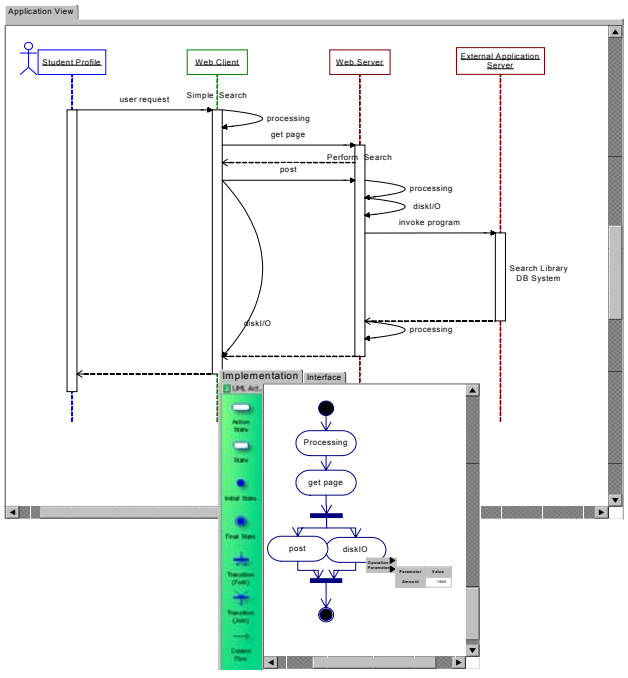

Figure 3: Activity Diagram

User behavior is described through *user profiles* activating *web clients*. Each profile includes *user requests*, which invoke specific *components* of a *web client* operating on the user workstation. Each *user request* acquires an *activation probability* attribute, indicating how often the user activates the specific application (as applications are composed of specific tasks, modeled as *process components*). The user profile concept may adequately represent user behavior when it can be predetermined. The behavior of Intranet users usually acquires such characteristics. On the other hand, Internet user behavior is ambiguous. In this case, user profiles may adequately represent the behavior of user groups requiring specific services, such as clients paying their credit card using a web banking system. D*aemon profile* models represent the automated activation of processes, and operate on the same processing node as the process they activate. Profile models are similar with process models.

3.3. Site View

Defining the access points of the system is supported through the *site* concept. User profiles are associated with *sites* during the functional configuration stage within site view. Sites are composite entities that can be constantly refined. During the logical configuration stage, process and files replicas are placed within sites. Site refinement is related to network topology, as discussed in section 4. Sites are represented through *UML packages* containing the corresponding component diagram (figure 2). Within site view, user profiles, daemon profiles and processes are viewed as *UML components*. Only user and process interaction are depicted within the site view. Thus, only user requests and process component interfaces respectively are represented within site view. Both are modeled as *UML component interfaces*. Process activations are viewed as *UML dependencies*.

Process entity participates in multiple UML diagrams; thus, *process* class (figure 2) combines properties of different UML entities by multiple inheritance. Other EIS model entities, especially the ones participate in application description, have multiple representations as well.

3.4. Operation dictionary

Operations depict “simple” tasks occurring in the system, such as “get page from a Web Server”, “insert data in a database” and “store data in the storage device”. We propose an appropriate *operation dictionary* for application description. The dictionary overall includes:

a. operations indicating basic tasks. These are*: processing,* indicating data processing, *request,* indicating invocation of a server process, *transfer,* indicating data transfer between processes and *synchronize,* indicating replica synchronization.

b. file related operations, involving File Server activation. These are *write* and *read*, indicating data storage/retrieval. While *processing* is an elementary operation, *write* can be expressed through simpler ones, i.e. a *process* and a *request* sent to a *File Server*.

c. database operations depicting database functionality. There are: *insert, delete, update, select* and *activate_store_procedure*. They provide transparency when defining application functionality.

d. web-related operations used to describe web server and web client functionality, such as: *get/put page*: indicating retrieving/storing an HTML/XML page, *get applet:* indicating applet download and *invoke program:* indicating active program invocation.

Evidently, the term "simple" is rather vague, as even simple operations must be ultimately decomposed into elementary ones (i.e. *processing, storing* and *transferring*) to estimate the QoS required from the underlying network. Node elements are responsible for performing corresponding *elementary operations* (e.g. processing and storing). As the aggregate functionality of an application can be internally translated into these *elementary operations*, we may determine individual elementary operation characteristics and ultimately estimate the QoS that must be provided from network entities. *UML use-case diagrams* are used to model operation decomposition. In this case, operations are viewed as *UML use cases*. The Operation use case diagram includes three types of use cases:

a. *application* invoked by actors, which represent operation included in operation dictionary

b. *intermediate,* which represent intermediate operations used to simplify operation decomposition and

c. *elementary,* which correspond to elementary operations

Operation decomposition is depicted using the «*uses*» *relationship* between use cases. The semantics of the uses relationship were extended to include *invocation order* and *parameter value list* properties. The *parameter value list* contains values for all parameters of the invoked operation. A fraction of the operation dictionary is presented in figure 5. It is important for the system designer to further extend the operation hierarchy to describe the functionality of specific applications [14]. When defining a new operation, the system designer must add it in the diagram, specify its parameters using a popup window and connect it through an «*uses*» *relationship* with the existing operations involved in its description (figure 5). In this context, UML notation facilitates the automated implementation of model extensions.

4. The Site Concept

System access points definition and process/file replica placement is supported through the *site* concept. Site specification is performed at levels of increasing detail to enable the progressive refinement of site structure. At the first level of detail, sites are defined as Internet access points. At the next levels, each *site* is further refined, allowing the user to adjust the *site* description according to the topology of the actual site described (e.g. Campus, Building, Floor) and user distribution. As shown in figure 2, each site must be supported by a network, thus *site* definition is restricted by the same rules as network definition. Since site and network concepts are associated, a site should be decomposed into *sub-sites* until site range corresponds to the limits of a LAN (*simple sites*), although the user may choose to define a different site structure. The progressive definition of sites enables the progressive solution of resource allocation and network configuration problems, while sites may be split or merged to ease network topology design.

Logical and physical configurations are performed progressively for each site. Since network topology may not be predefined during resource allocation, it should be concurrently designed to ensure the efficient support of the solutions adopted. Since sites are composed by "sub-sites", logical and physical configurations are recursively invoked for sub-sites of the same-level of a given site, as indicated in the following:

- Resource allocation algorithms are invoked to place server and file replicas in sites of the same level.
- Networks support sites, while sites of the same level are interconnected (if needed) using internetworks. The parameters of Quality of Service (QoS) for data exchange between sites are estimated. Internetwork topology is defined based on the estimated QoS parameters and existing network infrastructure.
- Merging and splitting sites may be performed to improve performance and reduce implementation cost, based on empirical rules. For example, if two sites are hosting the same server and data file replicas, they are candidates for merging. Sites can be also merged if they can be supported by a single network to simplify network architecture. Sites are split to provide better QoS. If a site split occurs, the logical/physical

configuration of the specific level is repeated (e.g. the aforementioned steps are re-activated).

- For each site, the aforementioned steps are performed in the next level of detail.
- At the lowest level of detail (simple sites), process and file replicas are allocated to processing nodes. The architecture of the corresponding LAN is afterwards defined. Simple sites can be merged if they can be supported by a single LAN or split if better QoS is needed. In such a case, process and file replicas and the architecture of network nodes may also be redefined.

Basically, sites are used when defining application configuration. Introducing progressive site refinement and linking site range to network range, enables the identification of dependencies between application configuration and network topology. Thus, the performance of the proposed solutions may be more accurately predicted.

4.1.Logical Configuration Policy

The allocation policy aims at: a. ensuring application performance and b. minimizing complexity and communication cost. Complexity denotes the number of networks forming the communication infrastructure. Communication cost refers to the QoS required from internetwork connections The QoS provided by each network should be within the limits supported by widely used technology (e.g. Ethernet technology for LANs) to ensure that the proposed solutions can be implemented within affordable price limits. Load balancing is also taken into account. Minimizing the communication cost between *sites* in a network environment can be described by the function of figure 4. As proved in [15], minimizing this function is NP-complete. To reduce the solution space, when dealing with client-server application the assumptions presented in the same figure are valid.

$$\min \sum_{c\in Pc}\sum_{\substack{i,j=1\\ i\neq j}}^{N} u_{ij}^{c}c_{ij}^{c} + \sum_{s\in Ps}\sum_{\substack{i,j=1\\ i\neq j}}^{N} p_{ij}^{s}c_{ij}^{s} + \sum_{f\in F}\sum_{\substack{i,j=1\\ i\neq j}}^{N} f_{ij}^{f}c_{ij}^{f} \quad (1), \text{ where}$$

- N is the overall number of sites
- c_{ij}^{x} represents the communication cost when accessing a replica of process or file x residing in site j from site I
- $u_{ij}^{c}, p_{ij}^{s}, f_{ij}^{f} = {}^{1}_{0}$, depending on whether a replica of process c or s and file f residing in site j is accessed by site I

Assumptions

1. A workstation is allocated to each user running a web client program, thus $\sum_{c\in Pc}\sum_{\substack{i,j=1\\ i\neq j}}^{N} u_{ij}^{c}c_{ij}^{c} = 0$

2. Web server replicas are allocated in all sites. Server replicas are allocated in the same site as the files they directly access. File server replicas are located in all sites where files are allocated. Independent files (e.g. non-sharable and non-updatable) and the servers directly accessing them are allocated in all sites they are accessed from, thus equation (1) results in

   $\min(\sum_{s\in Psf}\sum_{\substack{i,j=1\\ i\neq j}}^{N} p_{ij}^{s}c_{ij}^{s} + \sum_{f\in Fs}\sum_{\substack{i,j=1\\ i\neq j}}^{N} f_{ij}^{f}c_{ij}^{f})$ (2), where P_{sf} is the set of servers accessing dependable files (e.g. sharable and updatable) and F_s is the set of these files. There is always dependency between the two terms of the equation.

3. Provided that no optimal solution is needed, one can assume that there is a sequence of files for which (2) can be represented as $\sum_{f\in Fs}\sum_{r\in Rf}\min(\sum_{s\in Pr}\sum_{\substack{i,j=1\\ i\neq j}}^{N} p_{ij}^{sr}c_{ij}^{sr})$ (3), where Rf is the set file f replicas and Pr is the set of processes accessing each replica. It is assumed that for each process s in Pr, a replica is allocated in the same site as r. c_{ij}^{sr} represents the communication cost when process s access replica r and is invoked by others to access it.

Figure 4: Resource Allocation Problem Representation

Different replication policies, for example master/slave configuration, can be explored [16]. Both synchronous and asynchronous replica synchronization may be supported. The replication policy supported for each file

is defined during functional configuration. When adopting asynchronous replica synchronization having relatively small communication cost, files are widely replicated. Before analytically calculating communication cost and exhaustively solving (3), empirical algorithms consisting of rules are applied to partially solve the problem, even if the optimal solution is not reached. For example, it is assumed that files are allocated before processes. The order in which files are allocated affects the proposed solutions. The files are ordered according to the number of sites and processes accessing them (less accessed are allocated first). If the proposed solution is not efficient, the file allocation order is modified.

5. Case Study

The proposed methodology was applied for the configuration of the integrated information system of the *Greek National Diabetes Network (GNDN),* formed by the National Diabetes Institute and 168 Medical Centers hosted in public hospitals. The information system supports the following services: a) medical record maintenance regarding diabetic patients, b) provision of statistical information concerning the diabetes disease, c) everyday life patient support and d) educating the public regarding the Diabetes disease. Application design and implementation was performed using the Oracle product suite. The size of medical centers differs according to the size of the hospital hosting it. In the following, we comment on our experience using the EIS model through configuration stages.

5.1. Application Description

Most applications were developed based on the "typical" Oracle web-based architecture, where application interface is implemented using Java servlets executed in the *Oracle Application Server*. Database-related application logic is implemented using stored procedures. The Application Server is mainly responsible for the invocation of forms, the management of fields and the completion of transactions consisting of stored procedures. Since a Web Server is incorporated within Oracle Application Server, it is able to accept and process HTTP requests produced by the Web clients. To indicate the advantages of the modeling scheme, we discuss *Medical Record* application as an example.

Since medical records are private, two different database servers were modeled (each one belonging to a different application): the *Medical DB,* maintaining medical records, and the *Informational DB,* maintaining specific record fields subjected to statistical processing. It is evident that the two databases had to be synchronized.

OracleApplSrv was modeled using *Web Server* entity (figure 2). Two new operations integrating web-based functionality were added in the operation hierarchy to ease *OracleApplSrv* description. The *form access* operation was added in the dictionary to depict accessing, activating and processing of a form (figure 5). This operation is further decomposed into *get page, post* and *processing* operations to depict the invocation of Oracle forms as HTML pages and the filling of specific fields. As indicated in figure 5, *form access* operation is added as a new UML use case, "using" (e.g. is decomposed to) five (5) existing operations. For every new operation added in the dictionary, its type (application or intermediate), parameters and decomposition scheme are defined. The type and parameters are defined through a popup window. Operation decomposition is depicted using the «*uses» relationship* between *use cases*. The *invocation order* is defined for each relationship. Parameter values passed through the *parameter value list* may be either constant values or variables defined as form access parameters (see *get page* operation invocation).

The *activate_transaction* operation was added to depict the activation of stored procedures corresponding to each transaction. Stored procedures are modeled as Database Server components. *Activate_transaction* is further decomposed into *invoke program, processing* and *post* operations.

The addition of custom operations in the Operation Dictionary was essential to group existing operations used to describe repeated functionality within application components. It simplifies component description and, at the same time, ensures detailed application functionality description. Within *Medical Record* application, *OracleApplSrv* process consists of six (6) different components. Each one of them is described by two to seven operations, while it can be decomposed to a large number of elementary operations (varying form 97 to 245).

The Application and Site View for Medical Record Application are depicted in figure 6. As indicated in the application view, physicians invoke a web client to process/search medical records. Custom operations *form_access* and *activate_transaction* are used in *WebClient* and *OracleApplServ* process component descriptions respectively.

Since the *Informational DB* is updated only through database replication, *synchronize* operation had to be implemented according to Oracle replication mechanisms. This operation corresponds to the invocation of *synchronize* component of a database server.

A daemon profile is used to model synchronization invocation parameters. During functional configuration, only user profiles are placed within sites in Site View.

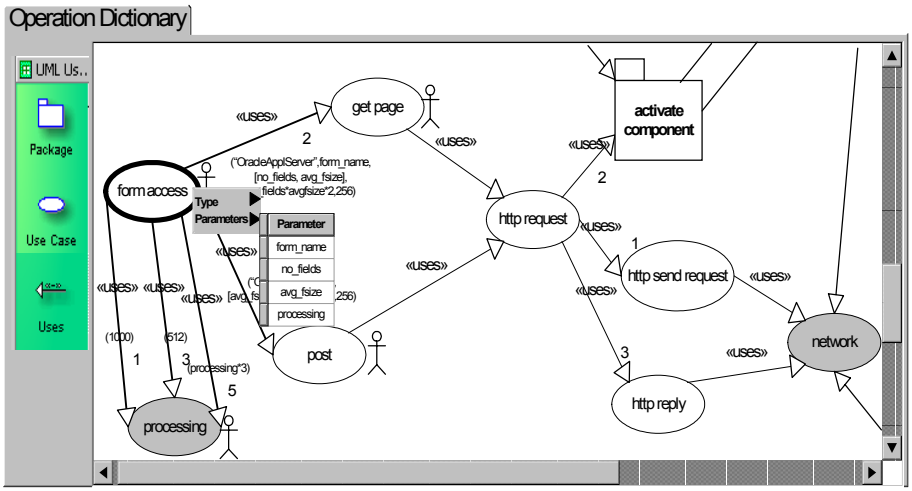

Figure 5: Definition of *form access* operation

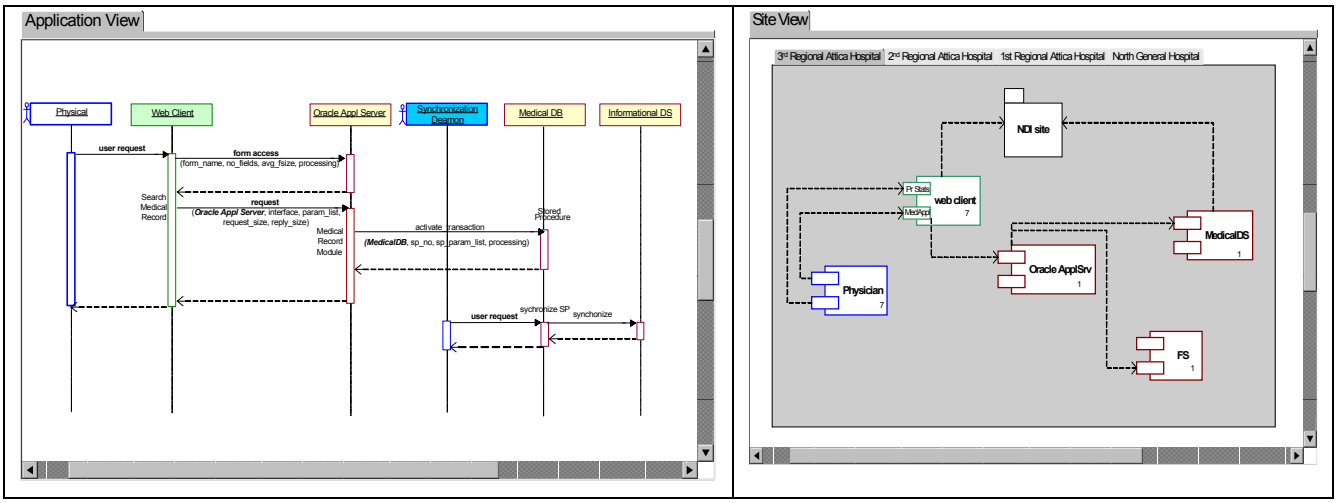

Figure 6: Application and Site View - Medical Record Application Model

5.2. System Configuration

NDI and medical centers are interconnected through the National Health Network, a private TCP/IP based network. Thus, physical specifications are partially predefined. During functional configuration, site description was conducted using two levels and the entire medical centers where placed in the second one. During logical configuration site decomposition hierarchy was modified to depict the structure of the WAN network interconnecting hospitals, thus it was feasible to consider the network topology when placing process replicas. A replica of Oracle Application Server and related files were placed in all medical centers to ensure performance. The implementation of an Oracle Database in small-sized Medical Centers is costly. Alternative *Medical DB* allocation scenarios were studied in order to ensure the requested response time and reduce cost. After evaluating different alternatives, it was decided to keep *Medical DB* replicas in all Medical Centers except for the small ones directly connected with another one with a network connection faster than 512Kbps. The process replicas placed in *3rd Regional Attica Medical Center* are depicted in figure 6 (site view). The number in the down right side

of each component represents the number of profile or process instances operating in the specific site.

GNDN network and database architecture could be modeled and studied using various commercial simulation tools. Due to the increased complexity of application functionality, e.g. Oracle Application Server, the direct mapping of application description into low-level primitives was not feasible. However, the proposed EIS configuration methodology and model proved to be efficient especially in accurately estimating QoS parameters and studying alternative process and data replication scenarios considering existing network topology.

6. Conclusions

We proposed an UML-like EIS model emphasizing the symbiotic relationship between application configuration and underlying network topology. Site refinement process (using merge and split) illustrates the ability of the proposed model to depict the impact of technological boundaries (physical specifications) to application functionality (functional specifications). The proposed model enables the exploration of dependencies between configuration stages even if they aren't obvious, since functional specifications are corrected or filled, as physical specifications are progressively defined. The extendable *operation dictionary* was proven to be an essential feature in order to describe complex application functionality, such as the one supported by Oracle Application Server, since it enables accurate application functionality description and the direct mapping of this description into QoS parameters that the underlying network must satisfy. This proved the main advantage of the proposed model. UML-like representation of model entities helped through model extension/customization, since a) system designers are familiar with UML constructs and b) automated code generation can be achieved.

7. References

[1] Serain D., *Middleware*, Springer-Verlag London, Great Britain, 1999.

[2] Oracle Co, *Oracle9i Applications Server Documentation Library Release 2*, December 2002.

[3] IBM Co, *Technical paper: Guide to the WebSphere Portal*, February 2003.

[4] Shedletsky J. and Rofrano J., "Application Reference Designs for Distributed Systems", *IBM System Journal*, Vol. 32, No 4, 1993.

[5] Savino-Vázquez N.N. et al., "Predicting the behaviour of three-tiered applications: dealing with distributed-object technology and databases", *Performance Evaluation* Vol. 39, no 1-4, Elsevier Press, 2000.

[6] Gomaa H., Menasce D., Kerschberg L., "A Software Architectural Design Method for Large-scale Distributed Information Systems", *Distributed System Engineering Journal,* Vol. 3, No 3, IOP, 1996.

[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] 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*, IEEE Computer Press, 2001.

[9] OMG Inc, *OMG Unified Modeling Language Specification*, Version 1.5, March 2001.

[10] Gomaa H. and Shin M., "Multiple View Meta-Modeling of software Product Lines", in Proceedings of the *8th International Conference on Engineering Complex Computer Systems*, IEEE Computer Press, 2002.

[11] Mirandola R, Cortellessa V., "UML Based Performance Modeling in Distributed Systems", *Lecture Notes in Computer Science 1939, UML2000*, Springer-Verlag, 2000.

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

[13] Nikolaidou M., Lelis D., et. al, "A Discipline Approach towards the Design of Distributed Systems", *Distributed System Engineering Journal*, Vol. 2, No 2, IOP, 1995.

[14] Nikolaidou M. and Anagnostopoulos D., "An Application-Oriented Approach for Distributed System Modeling", in Proceedings of the *21st International Conference on Distributed Computing Systems*, IEEE Computer Press, 2001.

[15] Morgan H.L., Levin K.D., "Optimal Program and Data Locations in Computer Networks", *Communications of ACM*, Vol. 20, No 5, ACM Press, 1977.

[16] Marreitti M, *Replication*, London, England, Academic Press, 1999.