Exploring Web-based Information System Design: A Discrete-Stage Methodology and the Corresponding Model

Mara Nikolaidou [1], Dimosthenis Anagnostopoulos [2]

[1] Department of Informatics and Telecommunications, University of Athens, Panepistimiopolis, 15771, Athens, Greece
mara@di.uoa.gr

[2] Department of Geography, Harokopio University
70 El. Venizelou Str, 17671 Athens, Greece
dimosthe@hua.gr

Abstract. After the enormous success of WWW platform, a great number of enterprise systems have web-based components. Although they are built using current technological treads, they often fail to provide the desired performance. A potential cause is that, design related problems, as application modeling, resource allocation and replication, network configuration and performance evaluation, although interrelated are solved in isolation. We, thus, argue that a concise methodology for effectively designing web-based information systems offers considerable capabilities. Four discrete stages, each of them addressing a specific issue, and their dependencies are identified. We also propose a common model for the representation of system entities throughout all design stages. UML-like notation was used as a visual tool for graphical representation of model components. Since the modelling scheme is extendable, the adaptation of UML constructs simplifies the process of extending or customizing the model. A case study where the proposed methodology was used for the design a complex enterprise system and the experience obtained are also presented.

1 Introduction

The World Wide Web (WWW) platform is often characterized as the middleware providing a common platform for Intranet-based and Internet-based application development [15]. Using a Web browser is possible to download that part of any application that consists of its user interface from both the Internet and the organization's Intranet. Such applications are considered as *web-based applications*, and are built based on the multi-tiered client-server model [12]. The first and second tiers, e.g. the user interface and the application service, is built using the WWW platform, while the other tiers implement the specific application logic that may be based on different architectures, as discussed in [14]. A *web-based enterprise system* can be described as a set of web-based applications and the underlying infrastructure (both at Internet and Intranet level). Most enterprise information systems built to support current technological treads are based on this architecture. Although significant vendors, as Oracle and IBM promote web-based software platforms during the last decade, the proposed solutions, although expensive, often do not provide the desired performance [13]. A potential cause is that, as most enterprise systems expand gradually, system extensions are performed without ensuring overall system performance. Furthermore, web-based applications are characterized by their internal complexity, the impact of which cannot be determined using a trivial mathematics. We, thus, argue that a systematic approach for effectively designing and evaluating web-based enterprise systems offers considerable capabilities [8].

Complete and accurate description of web-based applications is a critical factor in system design, since it ensures the accurate estimation of the Quality of Service (QoS) needed from the network infrastructure, the efficient allocation of resources and the efficient performance evaluation of the overall system. When configuring or evaluating network systems, applications are usually modeled as a series of discrete requests for processing, network transfer, etc., using predefined primitives [3, 9, 11]. We consider that such approaches lack efficiency to accurately describe web-based applications, since intermediate layers are required to support application decomposition in terms of multi-tiered client-server models and permit the accurate estimation of application load. The provision of higher-level primitives to easily describe standard web-based application tiers, as the user service implemented using the WWW platform, is also required. Extendibility of the model to facilitate the description of specific products must also be explored.

In the following, we present a methodology for web-based enterprise system design. It aims at providing decision making support to the system designer to ensure system efficiency when building a new system or extending an existing one. It includes web-based application package description, resource allocation and replication, network configuration (network topology design) and performance evaluation of the proposed architectures. Four discrete stages, each of them addressing a specific issue, and their dependencies are identified. When configuring web-based enterprise systems, each of these problems is usually handled in isolation, resulting in poor performance. It is important to note the significance of a common model for the representation of system entities throughout all design stages. We propose a meta-model supporting all stages incorporating the specific characteristics of web-based enterprise systems. This meta-model enables the exploration of dependencies between design stages even if they aren't obvious, since it is used as the reference framework to estimate application requirements, apply resource allocation and replication policies and construct network topology. Although the techniques explored may be generally applied for large scale systems, we emphasize web-based enterprise system design, as the application description model is introduced to efficiently describe web-based application functionality, while the resource allocation algorithms applied provide solutions taken into account the specific characteristics of web-based architectures. The proposed approach is supported by a set of software tools. The system designer interacts with them through a Java platform, named *Web-based enterprise system Modeler,* facilitating the graphical description of web-based architectures using the proposed meta-model.

Web-based enterprise system modeling is performed using UML constructs. The Unified Modeling Language (UML) is a graphically based object-oriented notation developed by Object Management Group (OMG) as a standard for describing software architectures, which gained widespread acceptance in the software industry [2]. In [4, 7], the UML language is used to model complex system functionality, while mathematical modeling, specifically queuing networks, is adopted to estimate application performance. Using UML sequence diagrams facilitates the complete description of client-server architectures, the triggering of processes and the information exchange between them [7]. However, the detailed description of process functionality is not facilitated. UML wide acceptance was the main reason why we chose to adopt it. UML-like notation was used as a visual tool for the graphical representation of web-based architectures. Since the modeling scheme is extendable, the adaptation of UML constructs to describe the web-based enterprise system meta-model helps the *Web-based System Modeler* user when extending or customizing the model.

The rest of the paper is organized as follows: In section 2, a design methodology for web-based enterprise systems is proposed. In section 3, the web-based enterprise system modeling approach is introduced and the benefits obtained during system configuration and performance evaluation are presented. In section 4, a case study employing the proposed approach and the experience obtained is presented, while conclusions reside in section 5.

2 Design Methodology

The configuration of web-based applications is performed based on the multi-tiered services. As described in [15], a typical web-based application architecture employed in numerous commercial solutions consists of the following:

a) Web client, i. e. the first tier, which facilitates a standard user interface allowing the user to retrieve information (in the form of HTML/XML pages) or activate applications (through HTML/XML page fields).

b) Web server, i. e. the second tier, which processes and redirects user requests, gathers results and sends them to the client in the form of HTML/XML pages. Thus, it provides a middleware platform integrating the desired functionality into the HTML/XML documents.

c) External application servers implementing specific application logic. Old-fashioned applications can be incorporated within the web environment using wrapping techniques. External application services can be activated through *CGI* programs at the web server site. The concept of a *context file* may be used at the server side, in order to temporary store the results of a CGI program before gradually presenting them to the user through the Web client. An alternative solution is the provision of a direct interface allowing the connection to an already active external program associated with a URL used for its invocation.

An alternative web-based architecture is the one based on intelligent web clients that support program execution. In this case, an *applet* may be downloaded from the Web server and be executed on the client machine to activate other tiers. The aforementioned functionality is depicted in figure 1.

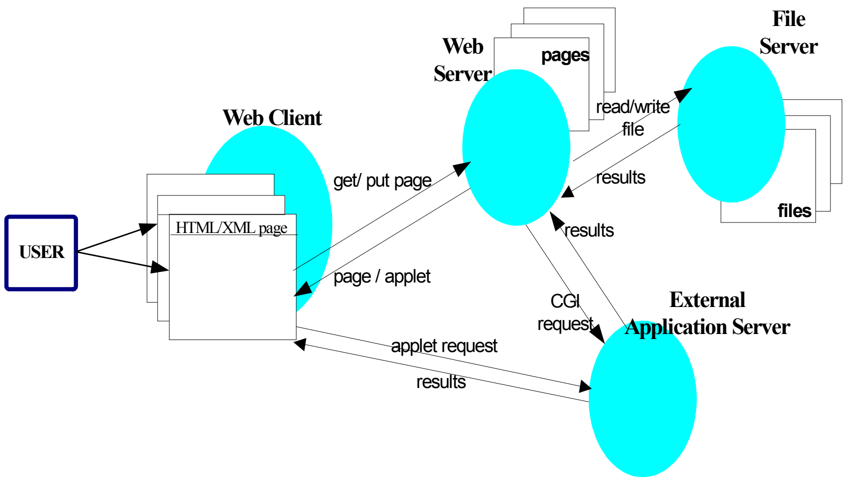


Figure 1. Widely-adopted Web-based Application Architecture

Web-based enterprise systems have the following specific characteristics:

- The first and second application tiers are implemented using Web technology. Since the first tier is only responsible for user interaction, application functionality is implemented as a set of services distributed on different servers. This reflects on service allocation policies, since a substantial part of application services must be close to the user.
- Replication techniques are employed to increase performance and availability especially over the Internet. 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. Replica synchronization is usually performed using asynchronous policies.
- Web-based applications usually operate on workstations. Users have their own workstation (diskless or not). Server processes are executed on dedicated servers machines. Application performance is greatly influenced by individual server machine performance.
- The communication between user-related tiers (web-based tiers) is based in HTTP protocol. Thus, identification of resources is accomplished using URLs or URIs. The network infrastructure consists of private intranets and Internet connections usually supporting TCP/IP protocol stack.

Based on aforementioned characteristics, we introduce a concise methodology for web-based enterprise system design. Four discrete stages and the dependencies between them are identified, as indicated in figure 2. Each of them addresses a specific issue explored during system configuration. All the stages are supported by a common meta-model used for web-based enterprise system description to ensure consistency.

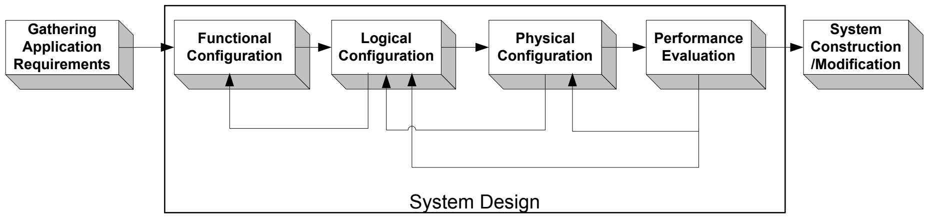


Figure 2. Web-based Enterprise System Design Methodology

Functional configuration (stage 1) corresponds to the description of system requirements (*functional specifications*). *Logical and physical configuration* (stage 2, 3) deal with process/data allocation and replication policies and network topology design respectively. Resource allocation and network configuration problems cannot be solved independently. Thus, stages (2) and (3) are invoked interactively until an acceptable solution is reached (*physical specifications*). It is important to note the significance of a common modeling scheme for the representation of web-based enterprise system entities throughout all configuration stages. Both logical and physical configuration stages are accomplished by properly instantiating specific properties of model entities, already defined during functional configuration. System configuration phase must facilitate the *performance evaluation* (stage 4) of the proposed solution prior implementation. If the system requirements are not satisfied, logical and physical configuration must be repeatedly invoked.

To support design stages, we have implemented a software platform written in Java. It is named *Web-based System Modeler* and facilitates a) the graphical interaction with the system designer for describing the functional specifications and exploring the proposed solutions using UML-like notation, b) the instantiation of the configuration stages and c) the interaction with specific software tools supporting each stage. Functional configuration is performed using the Modeler. Logical and physical configurations are accomplished using heuristics. IDIS [10] is a knowledge-based system facilitating the representation and exploration of resource allocation and network topology design algorithms using rules of thumb. IDIS knowledge bases were extended to support web-based application functionality and the proper interfaces were developed, so that the Web-based System Modeler may use IDIS to explore resource allocation and network design problems. To evaluate system performance, a discrete event simulation tool was used [9]. Object-oriented modeling and pre-construction of model components were employed to ensure efficiency [1]. System performance cannot be partially estimated, e.g. even if only a small part of the overall architecture is altered, the entire system must be simulated again to accurately estimate performance measurements. The completion of the simulation phase is the most time consuming part of the overall design phase. Thus, redesigning an inefficient architecture imposes the all possible changes are made before the simulation process is reactivated.

Web-based System Modeler consists of a graphical interface facilitating web-based enterprise system description using UML constructs, a dictionary containing models and restrictions and a set of wrappers for properly initializing external software modules and facilitating data exchange using object-oriented representation.

3 Modeling Approach

Web-based enterprise systems can be modeled as a set of interacting components, composite or not, suited to describe specific system functionality [5]. The level of detail in component description must ensure its accuracy and completeness. Thus, a multi-level modeling schema must be introduced providing: a. high-level composite models enabling the designer to go through configuration stages and b. a consistent method for the analysis of composite models in to elementary ones depicting simple network-based operations. Since the proposed approach is focused on web-based enterprise system design, high-level models must explicitly depict standard web-based application functionality, allowing the designer to describe the system under study even if not being aware of specific implementation details, such as HTTP operation.

We argue that it is efficient to use a common model to depict the desired functionality through all the stages of web-system configuration, since it significantly contributes to simplifying the overall process. The model must enable the description of both the *functional specifications* (e.g. application logic and user behavior) and the *physical specifications* (hardware infrastructure). The functional specification and parts of the physical specifications (if pieces of the hardware or network infrastructure are already available during system configuration) are defined during functional configuration stage. The logical configuration consists of defining the relationships between functional and physical specifications, since resource allocation and replication policies result in the allocation of processes and data instances to hardware components. Physical configuration results in the creation of physical specifications. The meta-model introduced to define web-based enterprise system functionality is presented in figure 3 using UML notation. Gray rectangles represent first level entities. It identifies a basic set of object types to describe functional and physical specifications and the relations between them. Further object types may be added by the designer to describe additional functionality by extending or restricting existing object behavior. Meta-model extension is essential in order to enrich the model capability to describe custom applications.

The physical specification consists of a multi-level network architecture. Each *network* either consists of other *networks* and an *internetwork,* describing the way they are connected to each other, or represents a simple LAN or WAN connection. Network *nodes* are either *workstations* allocated to users or *server* stations, where server processes operate. Networks and internetworks also include *relay nodes* depicting routing/switching functionality. Each network/internetwork also includes a *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 [5].

Processing nodes consists of a *processing*, *storage* and *communication* element, while *relay nodes* consist of a *processing* and many *communication elements*, one for each network they relay. Since network modeling is widely explored in both research and commercial tools, we do not further discuss this issue [6]. The approach used for modeling network, relay node and communication element entities in our model is the one presented in [1]. Each node element is responsible for serving a corresponding elementary operation. Consequently, application functionality should be internally translated into *elementary operations*. Based on elementary operation characteristics, we can determine the QoS provided by the physical specifications.

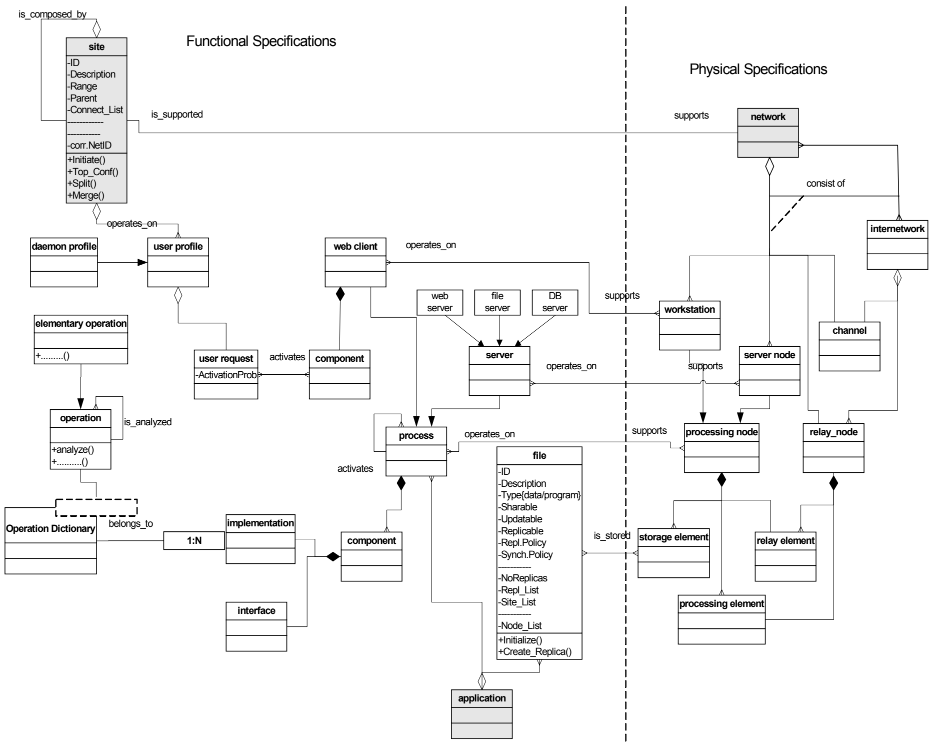


Figure 3. System Description Meta-model

The functional specification consists of web-based application and user behavior description. Web-based applications are presented as sets of interacting *processes* and *files* accessed by them. User behavior is described through *user profiles* activating *web clients*. Each profile includes *user requests* resulting in application invocation through the web interface, e.g. by invoking specific *components* of a *web client* operating on the user workstation. Each *user request* is characterized by an *activation probability* attribute indicating how often the user activates the specific application to perform a specific task modeled as *component*. The user profile concept may efficiently depict the behavior of a specific user if this can be predetermined. The behavior of Intranet users usually acquires such characteristics. Internet user behavior is more stochastic. In this case, the user profile concept may be used to depict the behavior of user groups regarding specific services, as for example, the user profile "remote client", which can be used to represent the behavior of clients paying their credit card using a web banking system. Daemon profile models indicate the automated activation of processes. Daemon profiles operate on the same processing node as the process they activate.

The services composing application functionality can be described using the *process component* concept. Each component corresponds to the set of tasks completed when activated in a certain manner. Besides the *web client process* model, *server process* models are also supported. Specific models as the *Web Server, Database Server* and *File Server* are introduced as ancestors of the basic server model to depict the corresponding functionality. Each process consists of many components, while each component consists of an *interface* depicting the process activation mechanism and an *implementation* comprising the tasks that occur upon process activation. *Component implementation* is described using a predefined set of *operations* forming the *operation dictionary*. Operations are described by qualitative and quantitative parameters, as for example, the processes being involved and the amount of data sent and received in an "invoke process" operation. In most cases, component implementation is executed sequentially (each operation is performed when the previous one is completed). However, there are cases where operations must be performed concurrently. All operations must be analyzed into elementary operations, namely *processing, storing* and *transferring*, to estimate the QoS needed from the physical specification. Operations depict simple tasks occurring in the system, for example "get page from a Web Server", "insert data in a database", "store data in the storage device". It is evident that the term "simple" is relevant in terms of application perspective. When describing a database, the operation "insert data" seems as simple, while describing a middleware platform seems as complex. In the following, we suggest an *operation dictionary* suitable for web-based application description. The dictionary includes:

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

b. file related operations involving File Server activation. These are: *write/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. They include:
   - *Get/put page*: indicating retrieving/storing an HTML/XML page
   - *Post:* indicating form/field passing on an HTML/XML page
   - *Get applet:* indicating applet download

- *Applet:* indicating applet activation
- *CGI*: indicating CGI program activation
- *Invoke Program:* indicating active program invocation
- *Handle/Retrieve context file*: indicating context file creation or modification/retrieval of context file data
- *HTTP request*: indicating send request/reply protocol implemented to support HTTP protocol

These *operations* are used to easily describe *component implementation* corresponding to Web server and Web client functionality in the model depicted in figure 1. When a *get page* request is sent to the *Web server*, the proper functions are initiated and a *HTML file* is retrieved from the *File Server* through a *read* request. Based on HTML page content, the Web server may send the *HTML page*, as a reply, back to the client, or initiate a CGI script or an active program to communicate with any *external application server*. The *get page* request is also used to download an *applet* from the *Web server* and communicate with the *external application server*.

Operations can be either elementary or of higher layer. In the latter case, they are analyzed into elementary ones. Operation decomposition is performed through intermediate stages to simplify the overall process and maintain relative data. Operation decomposition hierarchy ensures consistency, reduces complexity and enables following a common predefined decomposition mechanism. The most promising feature of this scheme is that the operation hierarchy can be further extended to include new operation, placed at the highest layer. Definition of new operations is based on existing ones to ensure consistency. Web related operation hierarchy is depicted in figure 4 using UML class diagram notation (only the decomposition of web-related operations is presented in the figure).

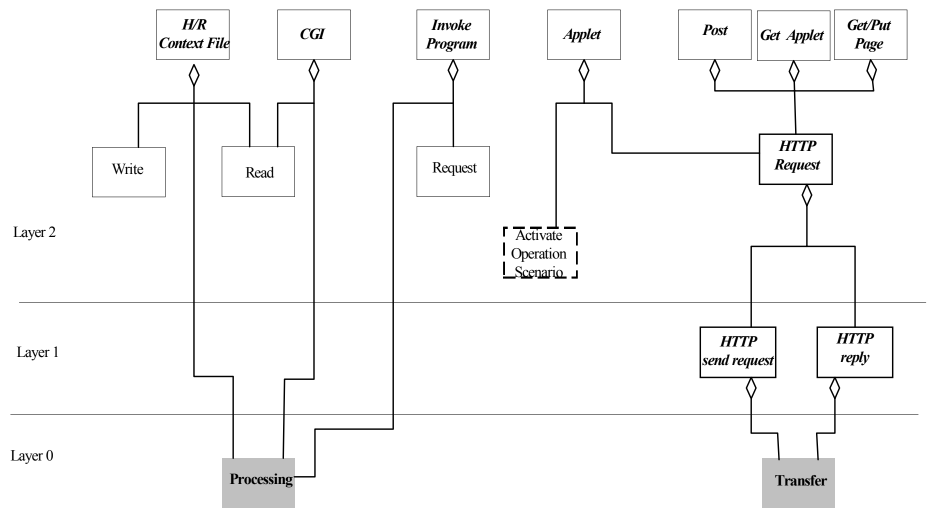


Figure 4. Web-related Operation Decomposition Model

Dotted rectangles represent *intermediate* actions, while gray rectangles represent *elementary* ones. Although not indicated in the figure, the *activate operation scenario* operation may result in the invocation of any operation included in application description. The *http request* operation depicts the request functionality as it is implemented by HTTP protocol and it is used in the decomposition of application operations describing *Web client* functionality. Many application operations, such as *get/put page*, actually represent the invocation of the corresponding server interface, and are decomposed into *http request* operation. These types of operations are supported to simplify the description of operation scenarios, since they are described using less parameters than the corresponding request operations. Furthermore, they make server invocation transparent to the user, when describing client operation.

Using *Web-based enterprise system Modeler*, the user may further extend the operation hierarchy to describe specific application functionality by properly extending the corresponding UML diagram. When defining a new operation, the user must specify its parameters and the operation used to describe it. In order to avoid model inconsistency, the user ability to add operations is restricted. The meta-model of figure 3 can be extended in a similar manner. We decided to adopt UML notation to describe the meta-model, as a. it is a wide acceptable standard and most designers are familiar with it, b. it allows the graphical representation of specifications and c. it facilitates the automated implementation of existing model extensions.

Site concept is introduced to define the access points of the system. Definition of *sites* is retrospective. Specification of their size is performed with respect to the user's view. At the first level of detail, sites are defined as Internet access points. At the next levels of detail, each *site* is further refined, allowing the user to adjust the description of the system according to the application scale. Since each site must be supported by a network, the *site* definition is restricted by the same rules as network definition; for example, a site should be decomposed to smaller ones until its range corresponds to the limits of a LAN. During the logical and physical configuration, sites of the same level may be merged or divided to ease network design. Progressive refinement of sites enables the progressive solution of resource allocation and replication problems.

Within Modeler environment, the user defines the system through a graphical environment. Each entity is depicted using a UML symbol, for example, the *UML package* symbol is used to depict composite entities as sites, applications and networks. All entities regarding their representation obtain attributes filled manually or automatically. The entity representation symbols are presented in table 1. The system offers two different views: the functional, emphasizing on the functional specifications, and the physical, emphasizing on the physical specification, respectively.

After completing the functional configuration stage, application description and site specification are completed. User profiles are defined within sites. Physical specification may also be partially defined. The next stage, e.g. logical configuration, corresponds to allocating processes and data forming applications into sites.

Table 1. Model Entity Visual Representation using UML Symbols

Model Entity	UML Representation
Functional Specification	
Site	⇒ Package
User/Daemon Profile	⇒ Component in Component Diagram
User Request	⇒ Interface of component
Process	⇒ Component in Component Diagram (site view) or Object in Sequence Diagram (application view)
Component	⇒ Interface of component or Object Activation (site view) in Sequence Diagram (application view)
Component Interface	⇒ Class
Component Implementation	⇒ Parameterized Class
Application	⇒ Package
Physical Specification	
Network/Internetwork	⇒ Package
Node	⇒ Nodes in Deployment Diagram
Element	⇒ Node within Node in Deployment Diagram
Channel	⇒ Arc between Nodes in Deployment Diagram

4 Case Study

The proposed design methodology was tested during the configuration stage of the enterprise information system of the *Greek National Diabetes Network (GNDN),* which consists of the National Diabetes Institute and 128 Medical Centers hosted in public hospitals allover the country. The National Diabetes Institute received a grant for building an integrated information system supporting 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. The system should be accessed through the Internet in order to provide information to different categories of users (public, patients, researchers) with different privileges, while it is maintained by physicians working in diabetes medical centers and the National Diabetes Institute.

Application design and implementation was performed using the Oracle product suite, while all applications should be web-based. Medical records are private, thus records belonging to a specific Medical Center should be only accessed from the personnel employed in it, while specific fields regarding clinical measurements can be widely accessed and statistically processed. The system supports almost 1.000.000 patient records. The size of medical centers differs according to the size of the hospital hosting it. There are three categories of medical centers regarding information system support, as indicating in table 2. As none of the applications is considered as time- critical, the required response time is 15-20 sec for all transactions. The National Diabetes Institute and medical centers are interconnected through the National Health Network, a private TCP/IP based network, interconnecting public hospitals. Network connection speed varies from 128 Kbps to 2 Mbps. In the following, we discuss detailed experience using the web-based enterprise system meta-model presented in section 3 through all the system configuration stages.

Table 2. Medical Center Requirements

Category	**Avg. Number of patients per day**	**Max. Number of Users**	**Number**
Small	<10	1	60
Medium	25	3	76
Lange	>45	7	32
Total	*4000*	*512*	*168*

Functional configuration

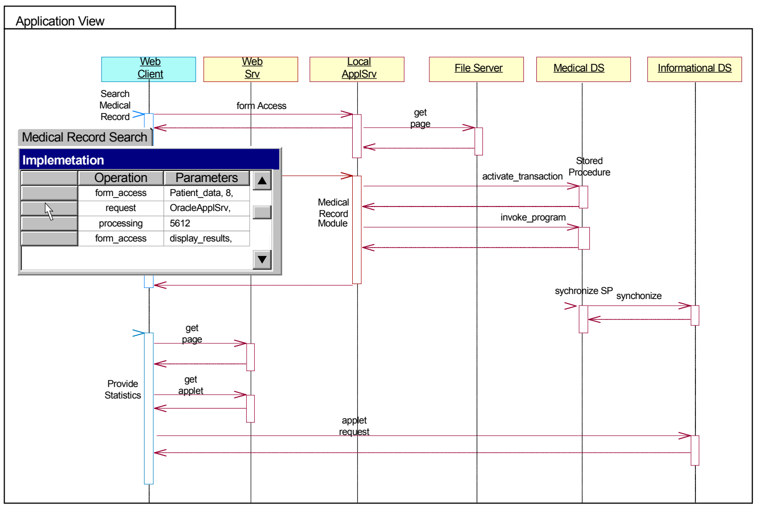
We focus on *Medical Record* and *Statistics Provision* applications, to indicate the advantages of the application modeling scheme. The Medical Record application is based on the "typical" Oracle web-based architecture, where application interface is implemented using JAVA servlets executed in the *Oracle Application Server*. Database-related application logic is implemented using stored procedures. Since medical records are private, two different database servers had to be modeled (each one belonging to a different application), the *Medical DS* maintaining medical records and *Informational DS* maintaining specific record fields subjected to statistical processing. It is evident that the two databases had to be synchronized.

OracleApplSrv was added in the metamodel depicted in figure 3 as an ancestor of *Web Server* entity to model Oracle Application Server functionality. *OracleApplSrv* mainly consists of 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 used by the users. Two new operations were added in the operation hierarchy (figure 4) to integrate web-based functionality and ease *OracleApplSrv* description. The *Form_access (form_name, numbers_of_fields, avg_field_size, processing)* was added in the operation hierarchy to depict accessing, activating and processing of a form. 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. The *activate_transaction(LocalDb, sp_number, [sp_list], processing)* operation is used to depict the activation of stored procedures corresponding to each transaction. It is further decomposed into *invoke program, processing* and *post* operations. The *invoke_ program(MedicalDS, call_stored_procedure, [sp_number, sp_list])* operation depictes the invocation of *MedicalDS* to execute a specific stored procedure, while the *post* operation is used to pass stored procedure parameter values. The *invoke_program(MedicalDS, insert, [512, 128])* operation represented recording the specific user performing medical record update and is included as the last operation of the 6 components of *OracleApplSrv* built to represent *Medical Record* application modules.

Extending operation hierarchy is an important feature to ensure the detailed application description. If only predefined operations could be used, the same description would have to be repeatedly given a) within the same application component and b) in different components, to represent the same functionality. Although only up to 15 operations are needed to describe an application module, each one of the 6 components of *OracleApplSrv* is decomposed to a large number of elementary operations varying form 97 to 245.

A new action called *call_stored_procedure_step* (*preprocessing*, *data_accessed, postprocessing)* was also added in the action hierarchy to easily describe store procedure functionality. Each stored procedure consists of one to five steps. The *call_stored_procedure_step* action includes the activation of *processing*, *read* and *write* actions. Since the *Informational DS* is updated only through database replication, *synchronize* operation scenario had to be implemented according to Oracle replication mechanisms. This operation scenario is invoked whenever *synchronize* action is invoked. An instance of Application Package View describing the discussed applications is depicted in figure 5.

As indicated in table 1, in the *application view* of functional specification, process functionality is depicted as sequence diagrams. Processes are represented as objects and components as object activations. When the user clicks on an activation icon, a popup window facilitates the description of its interface and implementation. Message icons are added automatically between process activations to represent process interaction. Messages are labeled using the name of the operation initiating process activation.



	Operation	Parameters
	form_access	Patient_data, 8,
	request	OracleApplSrv,
	processing	5612
	form_access	display_results,

Figure 5. Application Package View - Medical Record and Statistics Provision Application Models

As indicated in figure 5, the *Process Statistics* component invokes the WebSrv through *get_page* and *get_applet* actions and *InformationalDS* through *applet* action. It implements *Statistics Provision* application, a typical Internet application, where, after the user is identified, an applet is downloaded and executed in his/her workstation allowing the provision of specific statistics concerning the values of medical variables of patients fulfilling predetermined constraints.

Logical Configuration

The users invoke application modules through forms executed on the *Oracle Application Server* using a web browser, thus a replica of Oracle Application Server and related files were placed in all medical centers. The implementation of an Oracle Database in small-sized Medical Centers is costly. Alternative *Medical DS* 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 DS* replicas in all Medical Centers except for the small ones directly connected with another one with a network connection faster than 512Kbps. It was decided to place one copy of the *Informational DS* in the National Diabetes Institute, since the Internet access of the system is supported through this specific site.

Within functional specifications, 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 database replicas. An instance of the site view of functional specifications after completing logical configuration is depicted in figure 6.

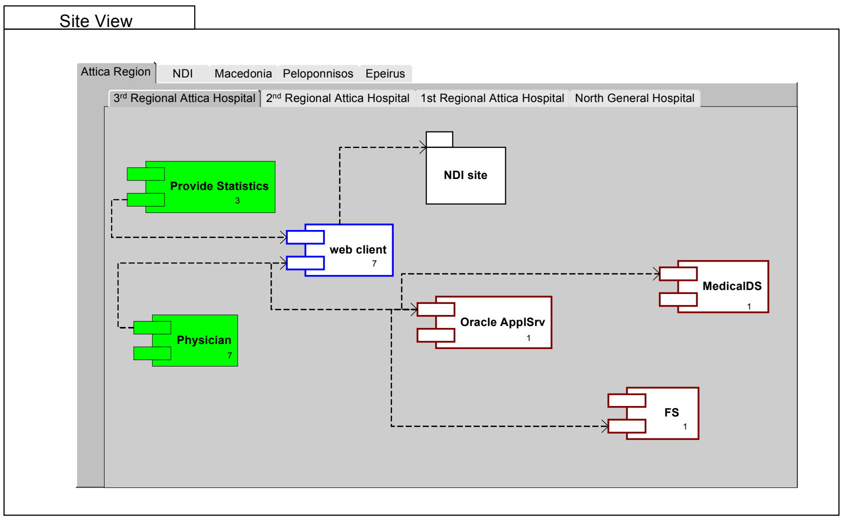


Figure 6. Site Package View – 3rd Attica Regional Hospital

As indicated in figure 6, site decomposition is depicted using package icons. Component icons are used to represent user profiles, daemon profiles and process (they are distinguished by their color). Interfaces represent user_requests, daemon_requests and components, respectively. The number in the down right side of each component represents the number of profile or process instances operating in the specific site.

Physical Configuration

Since the system uses the National Health network, the network topology was predefined. The physical configuration focused on the design of the National Diabetes Institute site. The performance evaluation of the system indicated that the processing power of the hardware supporting Database functionality was not adequate to execute client transactions within the predefined response time. This was mainly due to the lack of statistics preventing the estimation of Internet traffic characteristics.

The underlying network and database architecture could be modeled and studied using various commercial simulation tools. However, in order to accurately estimate the network load generated and study alternative database replication scenarios, there is a need for the detailed description of the applications. Due to the increased complexity of application functionality, e.g. Oracle Application Server operation, the direct mapping of application description into low-level primitives was not feasible. The extendable *operation dictionary* concept provided a set of common high-level constructs, conforming to web-based application specifications, which enable accurate application package description and the direct mapping of this description into QoS parameters that the physical configuration must satisfy. This is the main advantage of the proposed meta-model.

Site redefinition process illustrates the ability of the proposed model to depict the impact of technological boundaries (physical specification) and resource allocation policies to application functionality (functional specification). When configuring web-based enterprise systems, each of these problems is usually handled in isolation, resulting in poor system performance. The proposed model enables the exploration of dependencies between configuration stages even if they aren't obvious, as functional specifications are corrected or filled, as physical specifications are progressively defined.

5 Conclusions

We proposed a concise methodology for the design of web-based enterprise information systems considering their specific characteristics. Four discrete stages, each of them addressing a specific issue, and the dependencies between them were identified. They include application package description, process/data replica allocation, network/hardware configuration and performance evaluation of the proposed architectures. When configuring web-based enterprise systems, each of these problems is usually handled in isolation, resulting in poor performance. The common meta-model proposed for system description through all stages promotes consistency, since all the stages are performed by properly instantiating model entity properties. As indicated in the case study using the site concept, the model was able to represent how network/hardware technological boundaries effect functional specifications and system architecture even if they are not obvious.

The proposed meta-model facilitates the accurate and detailed description of web-based applications and the estimation of QoS parameters. Extending the meta-model was proven to be an essential feature in order to describe complex application functionality, such as the one supported by Oracle Application Server. UML-like representation of meta-model entities helped through model extension/customization, since Web-based System Modeler users are usually familiar with UML constructs.

References

[1] Anagnostopoulos, D.: An Object-Oriented Modeling Methodology for Dynamic Computer Network Simulation. International Journal of Modeling and Simulation **21** (2001)

[2] Booch, G., Rumbaugh, J., Jacobson, I.: The Unified Modeling Language User Guide. Addison Wesley (1999)

[3] Cruz, J., Park, K.: Towards performance-driven system support for distributed computing in clustered environments. Journal of Parallel and Distributed Computing **59** (1999)

[4] Kaehkipuro, P.: UML-Based Performance Modeling Framework for Component-Based Distributed Systems. In: Performance Engineering, Lecture Notes in Computer Science 2047, Springer-Verlag (2001) 167-154

[5] Kramer, J.: Configuration Programming - A Framework for the Development of Distributed Systems. In: Proceedings of the International Conference on Computer Systems and Software Engineering, Israel, IEEE Computer Press (1990)

[6] Law, A.M., McComas, M. G.: Simulation Software of Communications Networks: The State of the Art. IEEE Communications Magazine **4** (1994)

[7] Mirandola, R, Cortellessa, V.: UML Based Performance Modeling in Distributed Systems. In: UML2000, Lecture Notes in Computer Science 1939, Springer-Verlag (2000) 178-193

[8] Nezlek, G.S., Hemant, K.J., Nazareth, D.L.: An Integrated Approach to Enterprise Computing Architectures. Communications of the ACM **42 (**1999)

[9] Nikolaidou, M., Anagnostopoulos, D.: An Application-Oriented Approach for Distributed System Modeling. In: Proceedings of International Conference on Distributed Computing Systems, Phoenix, Arizona, US, IEEE Computer Press (2001)

[10] Nikolaidou, M., Lelis, D., et. al: A Discipline Approach towards the Design of Distributed Systems. IEE Distributed System Engineering Journal **2** (1995)

[11] Ramesh, S., Perros, H.G.: A multi-layer client-server queuing network model with non-hierarchical synchronous and asynchronous messages. Performance Evaluation **45** (2001)

[12] Reeser, R., Hariharan, R.: Analytic Model of Web Servers in Distributed Environments. In Proceeding of the International Workshop on Software and Performance, Ottawa, Canada, ACM Press (2000)

[13] Savino-Vázquez, N.N., et al.: Predicting the behavior of three-tiered applications: dealing with distributed-object technology and databases. Performance Evaluation **39** (2000)

[14] Shedletsky, J., Rofrano, J.: Application Reference Designs for Distributed Systems. IBM System Journal **32 (**1993)

[15] Serain, D.: Middleware. Springer-Verlag London, Great Britain (1999)