Distributed and Parallel Databases, 17, 267–290, 2005

A Systematic Approach for Configuring Web-Based Information Systems

M. NIKOLAIDOU mara@di.uoa.gr
University of Athens, Panepistimiopolis, 15771, Athens, Greece

D. ANAGNOSTOPOULOS dimosthe@hua.gr
Harokopio University of Athens, 70 El. Venizelou Str, 17671, Athens, Greece

Recommended by: Ioannis Vlahavas

Abstract. Enterprise information systems consist of interrelated Intranet and Internet-based applications, thus the Web platform serves well as the middleware for their development. Using a Web Browser, it is possible to download the user interface of any web-based application from anywhere around the world, providing transparency in application implementation. A *web-based information system* can be described as a set of web-based applications and the underlying infrastructure (both Intranet and Internet). Although most information systems built to support current technological treads are based on this architecture, they often fail to provide the desired performance. A potential cause is that, configuration issues, although interrelated, are solved in isolation. As the underlying network topology strongly influences application configuration, the relationship between the resource allocation policy and network architecture should be explored. We, thus, argue that a systematic approach for the quantitative analysis, effective configuration and detailed performance evaluation of web-based information systems is required. Four discrete stages are identified. The consistent representation of system specifications throughout all configuration stages facilitates the exploration of their dependencies even if these aren't obvious. For this purpose, we propose a common meta-model, incorporating specific characteristics of web-based systems. UML-like notation was adopted for system specification representation. Two are the main advantages of the proposed meta-model: extendibility, facilitating the description of applications at different levels of abstraction, and consistency, ensuring the accurate estimation of the Quality of Service parameters imposed to the underlying network by the described applications. A case study where the proposed approach was used for configuring a complex web-based system and the experience obtained are also discussed.

Keywords: Web-based system configuration, Web-based application modeling, resource allocation, UML specification

1. Introduction

Modern enterprise information systems consist of interrelated Intranet-based and Internet-based applications, built on multi-tiered client-server models [24]. Some of their main characteristics are: (a) wide scale, as they operate upon a variety of network platforms, (b) complexity, as they consist of cooperating heterogeneous distributed applications (e.g, database applications, workflow systems, web services), and (c) extendibility, as they

This work was partly funded by Pythagoras program co-funded by the Greek Goverment (25%) and the European Union (75%).

gradually expand to satisfy evolving requirements. As end-users interact with multiple applications, it is important to provide a common "look and feel" to application interfaces. The Web platform serves well as a common access environment for applications operating within enterprise Intranet boundaries, while, at the same 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 [20] and IBM WebSphere [28], supporting both proprietary software application environments and standard J2EE architecture. They facilitate application integration and enable user access through a common interface using a web client. Such applications are called *web-based applications* [24]. 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 [26]. A *web-based information system* can be described as a set of web-based applications and the underlying infrastructure (both Intranet and Internet).

Though vendors actively promote information system development using the aforementioned software platforms, the proposed solutions, although expensive, often do not provide the desired performance [23]. A potential cause is that, configuration issues, although interrelated, are solved in isolation. Since the underlying network topology affects application configuration, the relationship between resource allocation policy and network architecture should be explored [7]. Quantitative analysis of application behaviour must thus be supported, and resource allocation policies must be explored taking into account the restrictions imposed by existing network infrastructure and available technology. Furthermore, as most information systems expand gradually, system extensions are performed without ensuring the overall system performance. We, thus, argue that a systematic approach for effectively configuring and evaluating web-based information systems offers considerable capabilities, providing decision making support to system designers when building a new system or extending an existing one [5, 7, 16].

Complete and accurate description of application functionality is a critical factor in web-based information system configuration, since it ensures the accurate estimation of the Quality of Service (QoS) imposed to network infrastructure and allows the efficient allocation of resources and the performance evaluation of the overall system. When configuring or evaluating network systems, applications are usually modelled as series of discrete predefined primitives for processing, network transfer, etc. [2, 18, 21]. The QoS parameters, such as network throughput, imposed by individual applications must be modelled in detail, so that it can be better explained, predicted and controlled. We consider that the aforementioned approaches lack efficiency to accurately describe web-based application functionality, since intermediate layers are required to support application decomposition in terms of multi-tiered client-server models to enable the accurate estimation of application load. The provision of higher-level primitives to easily describe standard web-based application tiers, such 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.

Web-based system configuration is a multidisciplinary issue, imposing the examination of a large number of alternative architectural solutions and resource allocation and replication scenarios [7]. Each configuration problem may be viewed as a search for an optimal combination of interacting components, which consists of hierarchical layers of other interacting components [4] (e.g., networks consist of other networks). Such a problem is usually NP-complete. When configuring complex, large-scale systems, experts rely more on experience than on theory-based calculations [8, 16]. Thus, heuristics are often applied to minimize the search space of the problem. Expert system research has often concentrated on the representation and manipulation of heuristic knowledge and its use in resource allocation, information system design and network configuration problems [3, 4, 8, 11]. For solving each problem, specific models, such as queuing networks or Petri-nets, are adopted to represent system specifications. Thus, extensive modelling requirements are imposed, as models depicting system specifications for interrelated problems, must be "synchronized" to facilitate information exchange. Models must support different granularity layers to depict web-based systems internal complexity, and thus model synchronization should be supported in a multi-layer fashion, while "similar" granularity layers should be supported by "synchronized" models.

To explore different scenarios, evaluation of the proposed web-based architectures must be facilitated. When dealing with complex systems, simulation is widely applied to estimate the performance characteristics of a given architecture [1, 10, 22, 23]. This approach is also adopted in this paper.

In the following, we present a methodology for web-based information system configuration, addressing all critical issues: application quantitative analysis, resource allocation and replication, network configuration (network topology design) and performance evaluation. The proposed methodology aims at providing decision making support to the system designer, which may experiment with various scenarios, analyze them and elaborate the best solutions prior to system implementation. Four discrete stages, each addressing a specific issue, and their dependencies are identified. It is, thus, of extreme significance to support a consistent model for the representation of system specifications throughout all stages. We propose a common meta-model incorporating the specific characteristics of web-based systems. This model enables the identification of the otherwise 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. Although the techniques explored may be generally applied for the configuration of large scale systems, we emphasize the configuration of web-based systems, as the proposed application specification model efficiently describes web-based application functionality, while the resource allocation algorithms applied provide solutions taking into account the specific characteristics of web-based system architecture. The proposed approach is supported by a set of software tools. The system designer interacts with them through a Java platform, named *Web-based System Modeler,* facilitating the graphical description of web-based architectures using the proposed meta-model.

The Unified Modeling Language (UML) is a graphical object-oriented notation developed by Object Management Group (OMG) as a standard for describing software architectures, which gained widespread acceptance in the software industry [19]. Since system designers are usually familiar with UML, as a process and data modelling language, it was decided to use UML-like notation to model all aspects of web-based system specification in a multi-layer fashion by integrating different diagram types [6]. In Kaehkipuro [9] and Mirandola [14], 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 behaviour should also be incorporated in the model and dependencies between applications and network infrastructure must be modelled. Thus, different system views must be provided. In the proposed model, UML constructs were embedded in *Web-based System Modeler* graphical environment to facilitate the representation of web-based system entities in a multi-layer fashion. Since the modelling scheme is extendable, the adaptation of UML notation facilitates the extension or customization of the model by the system designer.

The rest of the paper is organized as follows: In Section 2, a web-based system configuration methodology is proposed. In Section 3, a web-based system meta-model is introduced and the potential obtained during system specification and configuration are discussed. In Section 4, we introduce the *site* concept used to depict system access points. Sites can be constantly refined, facilitating incorporating network infrastructure restrictions in the application configuration process, and visa versa. In Section 5, a case study employing the proposed approach and the experience obtained is presented, while conclusions reside in Section 6.

2. Web-based information system configuration

As depicted in figure 1, a typical web-based architecture employed in numerous commercial solutions consists of the following [24]:

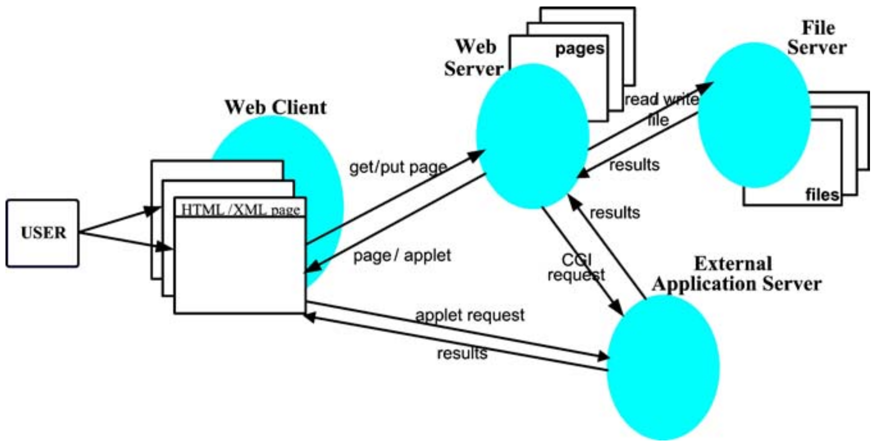


Figure 1. The WWW as a middleware platform.

(a) Web clients, i.e. the first tier, which provide 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 servers, i.e. the second tier, which process and redirect user requests to external application servers, gather results and send them to clients in the form of HTML/XML pages. Thus, the web platform acts as the middleware integrating the desired functionality into the HTML/XML documents. Communication between user-related tiers (web-based tiers) is based on HTTP.
(c) External application servers implementing specific application logic. External application services can be simply activated through *CGI* programs at the web server site. The provision of a direct interface allowing connecting to an already active external program is also commonly supported. Old-fashioned applications can be incorporated within the web environment using wrapping techniques. Identification of services and resources is accomplished using URLs or URIs.

An alternative approach is based on intelligent web clients that support program execution. In this case, an *applet* is downloaded from the Web server and executed on the client machine to activate external application tiers.

We introduce a configuration methodology for web-based systems based on the following assumptions:

- The configuration of web-based applications is performed based on multi-tiered services. 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 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.
- Users have their own workstation (diskless or not). Server processes are executed on dedicated server nodes. Thus, application performance is greatly influenced by individual server machine performance.

Four discrete stages and the dependencies between them are identified, as indicated in figure 2. *Functional configuration* (stage 1) corresponds to the description of system

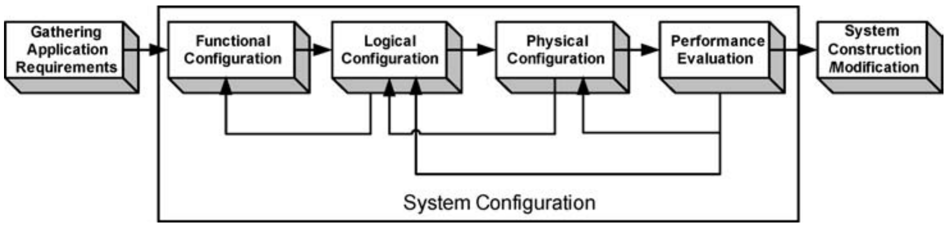


Figure 2. Web-based system configuration methodology.

requirements. *Logical and physical configuration* (stages 2 and 3) deal with process/data allocation and replication policies and network topology design, respectively. As resource allocation and network configuration problems must be jointly solved, stages (2) and (3) are repeatedly invoked until an acceptable solution is reached. System configuration phase must facilitate the *performance evaluation* (stage 4) of the proposed solution prior to implementation. If system requirements are not satisfied, logical and physical configuration are re-initiated.

To automate configuration stages, we have implemented a software platform, named *Web-based System Modeler*. It facilitates (a) the graphical interaction with the system designer for describing system 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. The Modeler is implemented in Java. Functional configuration is performed using the Modeler. Logical and physical configurations are accomplished using heuristics. For this purpose, IDIS [17] is used, 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 appropriate interfaces were developed, so that Web-based System Modeler uses IDIS to explore resource allocation and network design problems. To evaluate system performance, the discrete event simulator described in [18] is used.

3. Web-based system specification modelling approach

Web-based systems can be modelled as an aggregation of interacting components, either primitive or composite, usually customized to depict the functionality of specific systems. The level of detail in each component's description reflects the degree of accuracy and completeness.

We argue that it is imperative to use a common model throughout all stages of web-based information system configuration, as this contributes significantly to the consistency and simplification of the overall design. All configuration stages progressively construct this model, as indicated in figure 3. The system designer supervises system configuration through different model views, represented as UML diagrams.

System Specification consists of the *functional specification* (e.g., application logic and user behaviour) and *physical specification* (hardware infrastructure) of the web-based system, as follows:

1. Functional specification and parts of the physical specification (i.e. referring to existing pieces of computer/network infrastructure) are defined during functional configuration.
2. Logical configuration defines the relationship 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 specification.

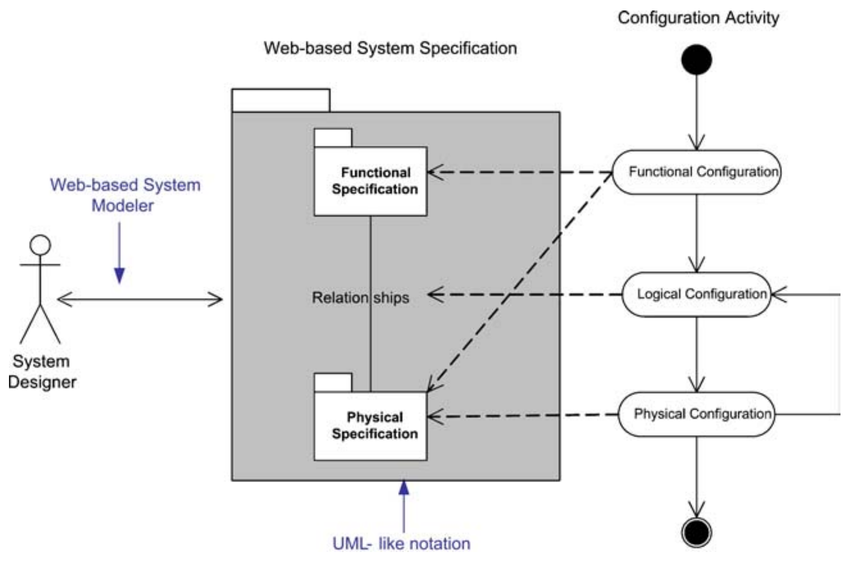


Figure 3. Web-based information system configuration framework.

The meta-model, introduced to describe Web-based system specification, is presented in figure 4. It identifies a basic set of object types to describe functional and physical specifications and their relationships. UML notation is used in the figure.

Gray rectangles represent first-level entities. Additional object types may be added by the designer to describe additional functionality by extending or restricting existing object behaviour. Meta-model extension is essential to enrich the model capability to describe custom applications.

The physical specification refers to the aggregate network architecture. Each *network* either:

- consists of multiple *networks* (1:N) and an *internetwork* interconnecting them, or
- represents a single LAN or WAN connection

Network *nodes* are either *workstations*, allocated to users, or *server* stations, running server processes. Networks and internetworks 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. Based on elementary operation characteristics, we can determine the quality of service (QoS) provided by the physical specification. 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. As network modelling has been widely explored in the literature [11], we do not further emphasize the modelling approach employed.

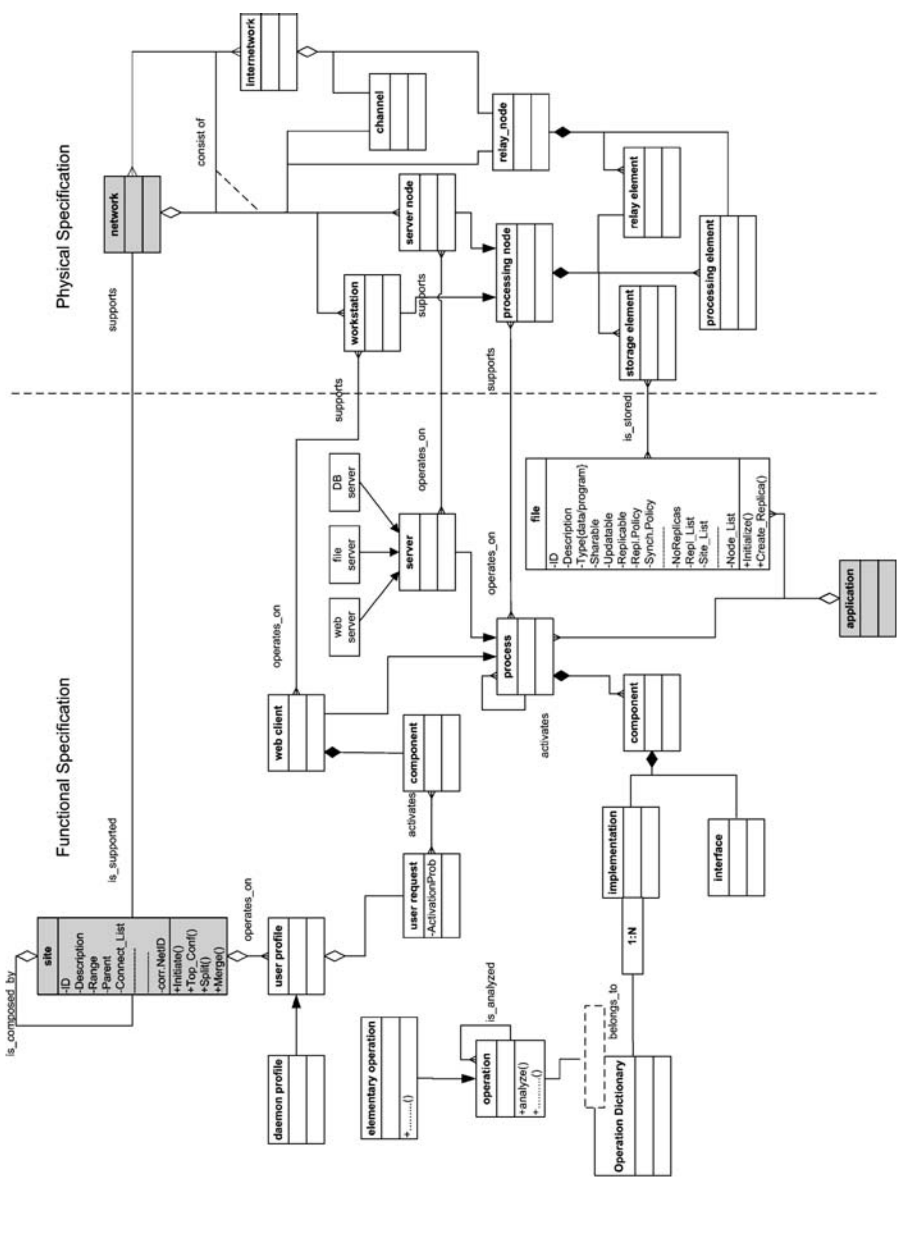


Figure 4. Web-based system specification meta-model.

3.1. Application modelling

Network modelling is crucial to determine if the network infrastructure is suitable for QoS required by web applications. An approach towards this objective can be based on the following conception: 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 the underlying infrastructure. This estimation is critical for the accuracy and completeness of physical specification. We, thus, emphasize application functionality modelling issues.

Functional specification involves the description of web-based applications and user behaviour. Web-based applications are conceived as sets of interacting *processes* and data repositories (i.e., *files)* accessed by them (as depicted in figure 4). The services composing the aggregate functionality of an application can be described using *process* and*component* concepts. As a process may in general 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 meta-model, a *process* is thus composed by the *components* corresponding to all alternative activation ways.

Components are composed of two distinct parts:

- *interface*, depicting the process activation mechanism and
- *implementation*, comprising the tasks performed upon process activation.

Component implementation is described using one or more operations from a predefined *operation* set*,* that is, the *operation dictionary*. Operations are described by qualitative and quantitative parameters, such as the processes involved and the amount of data sent and received in an “invoke process” operation.

To represent the first and second tiers of web-based applications, a *web client* and a *web server* model are incorporated in the meta-model (figure 4). The *web client* model is defined as an ancestor of the *process* model. The *web server* model is defined as an ancestor of the *server* model, which extends *process* model behaviour to implement basic server functionality. Models for *database* and *file servers* are also defined as ancestors of the abstract *server* model.

User behaviour 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 involve specific tasks, modelled as *process components*). The user profile concept may adequately represent user behaviour when this can be determined. The behaviour of Intranet users usually acquires such characteristics. On the other hand, Internet user behaviour is ambiguous. In this case, user profiles may adequately represent the behaviour of user groups requiring specific services, such as clients paying their credit card bill using a web banking system. Daemon profile models represent the automated activation of processes, and operate on the same processing node as the process they activate. Defining the access points of the system is performed using the *site* concept. User profiles are associated with *sites* during functional configuration.

3.2. Web 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". Evidently, the term "simple" is rather vague, as even simple operations must be ultimately analyzed into elementary ones (i.e. *processing, storing* and *transferring*) to estimate the quality of service required from the underlying infrastructure. We propose an appropriate *operation dictionary* for application description, involving the operations de facto considered as the simple ones in the web environment. The same has been adopted in other relevant approaches for network modelling, such as [11]. No operation can be referenced in application description when not previously modelled as an *operation dictionary* entry. Operations are described by parameters indicating qualitative and quantitative characteristics [18]. The dictionary overall includes:

(a) basic operations, indicating simple 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 *processing* 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.
- *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.
- *HTTP request*: indicating HTTP protocol calls.

These operations are used to easily describe web-based application functionality, as 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. A *post* request is sent from the Web client to indicate form field passing. Based on field content, the Web server may send a new HTML page, such as a reply, back to the client, or initiate a *CGI* script or an active program (*invoke program* operation) to communicate with any *external application server*. The *get applet* request is used to download an applet from the *Web server* and communicate with the *external application server* (*applet* operation).

When operations are of a higher layer, they are decomposed into elementary ones. Operation decomposition is performed through intermediate stages to simplify the overall process and maintain relative data. Such a standardized mechanism ensures consistency and reduces complexity, and is expressed as an *operation decomposition hierarchy*. Three types of operations are included in the hierarchy: *application* operations, included in operation dictionary, *intermediate* operations, used to simplify operation decomposition and *elementary* operations, used to estimate QoS parameters forced imposed at physical specification. An abstract view of the operation decomposition hierarchy, containing web-related operations, is depicted in figure 5. Operations are represented as objects in a UML class diagram.

Dotted rectangles represent *intermediate* actions, while grey rectangles represent *elementary* ones. Although not indicated in the figure, operations of different type, such as *request* or *read*, are also decomposed into simpler ones. The *activate component* operation results in the invocation of a process component, consisting of other operations. *Get/put page, get applet* and *post* operations represent the invocation of a corresponding *web server* component, thus they are decomposed into an *HTTP request* operation. Such operations simplify component implementation description, since they are described by less parameters than the corresponding request operations, while server invocation is transparent. The *request* operation represents a process component invocation and is further decomposed into *send request, activate component* and*reply* operations. The *HTTP request* operation depicts the functionality of request operation, as it is implemented by HTTP protocol, and it is used when describing *Web client* functionality.

It is important for system designers to further extend the operation decomposition hierarchy to describe the functionality of specific applications. Defining new operations is

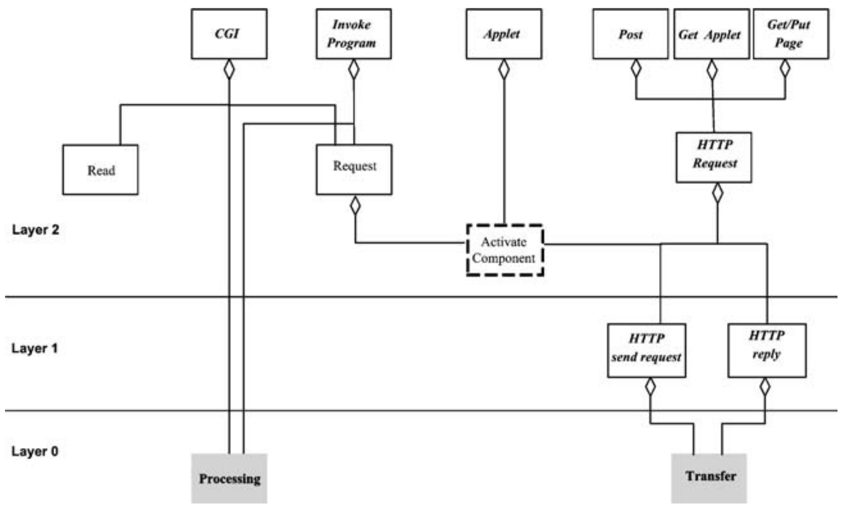


Figure 5. Web-related operation decomposition hierarchy.

based on the existing ones, to ensure consistency [18]. When defining a new operation, the operations it is decomposed into must be specified, along with their invocation parameters.

3.3. UML representation

UML-like notation was adopted for the meta-model description, as (a) it is a widely accepted standard, (b) it allows the graphical representation of specifications, and (c) it facilitates the automated implementation of model extensions. Web-based system specification is performed using the Web-based System Modeler environment. The Modeler offers three alternative views, the *application* and the *site views*, emphasizing functional specification, and the *physical view*, emphasizing physical specification. Different UML diagrams are used to represent each view. Each system entity is depicted by extending the properties of corresponding UML diagram entities, using the *stereotype* mechanism. All entities obtain attributes that are filled manually, during functional configuration, or automatically, during logical and physical configuration. UML representation of the meta-model introduced for web-based system specification is presented in Table 1.

Table 1. UML-like representation for Web-based system specification.

Model entity	UML representation
Functional specification	
Application view	*Sequence diagram*
Application	Package
Process	Object
Component	Object activation
User/Daemon profile	Object
User request	Object activation
Component implementation	Activity diagram
Operation	Activity/Message
Site view	*Component diagram*
Site	Package
User/Daemon profile	Component
User request	Interface of component
Process	Component
Process component interface	Interface of component
Physical specification	
Physical view	*Deployment diagram*
Network/Internetwork	Package
Node	Node
Element	Node within node
Channel	Arc between nodes
Operation decomposition hierarchy	**Class diagram**
Operation	Object
Operation decomposition	Composition relation

Application View consists of the supported applications, represented through UML packages containing the corresponding sequence diagram. Applications are conceived as sets of interacting *processes,* the data repositories (i.e. *files)* accessed by them and the *profiles* activating them. Processes and files are represented as stereotypes of *UML objects* and components as *object activations*. Component implementations are described using *UML activity* diagrams, supporting both sequential and concurrent operation execution. Operations are represented as stereotypes of *UML activity* entities. The system designer adds a new activity in the UML diagram and defines its properties, e.g., operation type and operation parameters, using a pop-up window. *UML message* entities are automatically added in the sequence diagram between process activations to represent process interaction. Messages are labelled using the name of the operation initiating process activation. User profile UML models are similar with process models. An example of the application view is depicted in figure 6, where a simple database search is initiated by a web user through the proper CGI in the Web Server. The UML activity diagram depicts the *Simple Search* component of *Web Client* process.

Sites are represented through *UML packages* containing the corresponding component diagram. 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 modelled as *UML component interfaces*. Process activations are viewed as *UML dependencies* (figure 9).

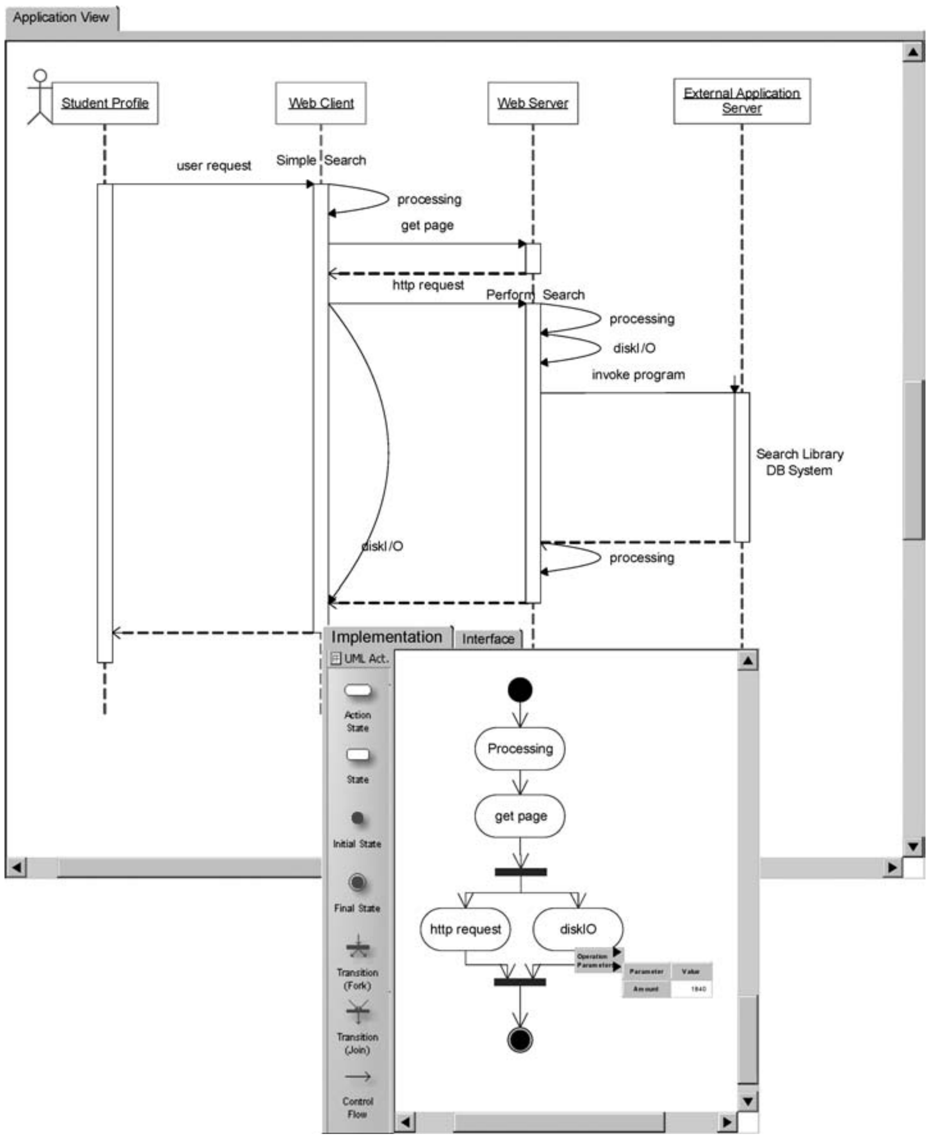
UML deployment diagrams are commonly used to represent network architectures [9]. In the physical view, networks and internetworks are represented as UML packages containing the corresponding deployment diagram. *UML arcs* represent channels. Network/internetwork nodes are represented as stereotypes 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. Node elements are also represented as stereotypes of *UML Node* entity.

When defining a new operation, the system designer must add it in the *Operation Decomposition Hierarchy,* represented as a UML Class Diagram. The designer must consequently specify its parameters using a popup window and connect it through the *consists_of* relationship with existing operations involved in its description. The *consists_of* relationship is a stereotype of the *composition* relationship, extending composition semantics to represent the invocation order and parameters of existing operations used to describe the new one (an example is given in figure 7).

4. The site concept

Defining access points is supported through the *site* concept. Site specification is performed at levels of increasing detail to enable the progressive refinement of site hierarchical structure. At the first level of detail, sites are defined as Internet access points. At the next levels, each *site* is further refined, allowing system designer to adjust the *site* description according to the topology of the actual site (e.g., Campus, Building, Floor) and the distribution of users.

As shown in figure 4, 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 the 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



Parameter	Value
Amount	1840

Figure 6. Application view—A simple example.

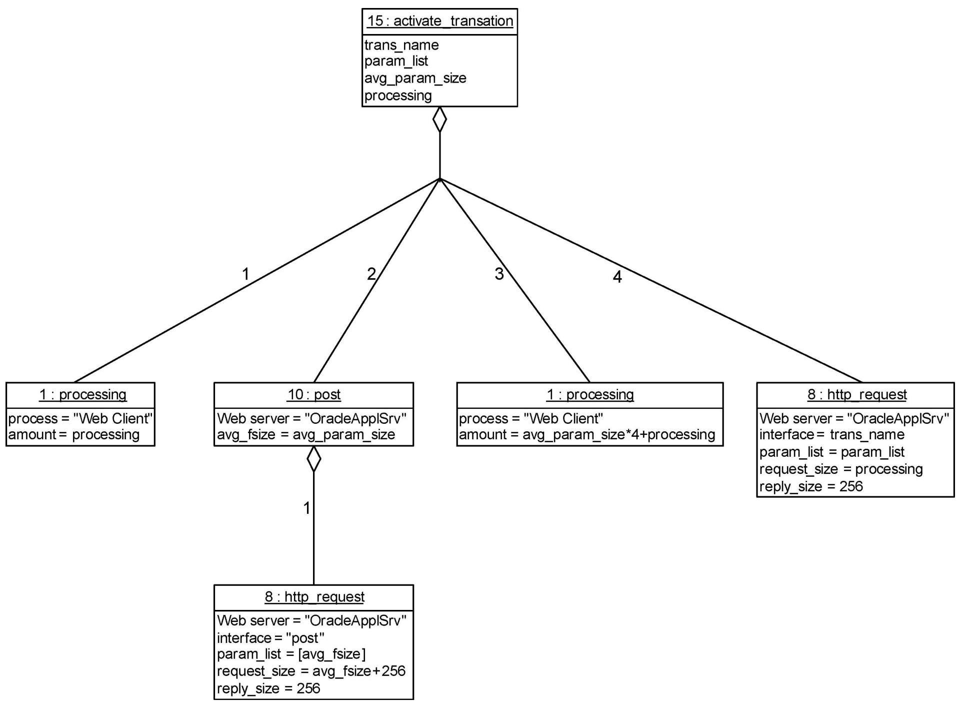


Figure 7. The *activate_transaction* operation decomposition hierarchy.

allocation and network configuration problems, while sites may be split or merged to depict network topology.

Logical configuration is 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. Thus, logical configuration performed for each site is followed by the physical configuration of the corresponding network. Networks support sites, while sites of the same level are interconnected (if needed) using internetworks. Since sites are composed by "sub-sites", logical and physical configuration stages are recursively invoked for sub-sites of the same-level of a given site, as follows:

1. Resource allocation algorithms are invoked to place process and file replicas in sub-sites of the same level.
2. Parameters of Quality of Service (QoS) for data exchange between sites are estimated. Internetwork topology is defined based on the estimated QoS and existing network infrastructure.
3. 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 also be merged if they can be supported by a single network, to simplify network architecture.

Site split provides a better QoS. When it occurs, the logical/physical configuration of the specific level is reconstructed (e.g., steps 1–3 are re-invoked).
4. For each sub-site, the aforementioned activities are performed at the next level of detail.
5. At the lowest level of detail (simple sites), process and file replicas are allocated to processing nodes. The architecture of the corresponding LAN is defined afterwards. Simple sites can be merged if they can be supported by a single LAN or a split may occur if a better QoS is needed. In such a case, process and file replicas and the architecture of network nodes may also be redefined.

The site concept is useful 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. Resource allocation 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 (for example, average data exchange rate) imposed to internetwork connections. The QoS imposed to each network is limited by widely used technology (e.g., Ethernet technology for LANs), since proposed solutions should be implemented within affordable price limits. Load balancing is also taken into account. Minimizing the communication cost between *sites* in a network environment is given in (1) (Table 2) [25]. As proved in [15], minimizing this function is NP-complete. To reduce the solution space, when dealing with client-server architecture, the assumptions presented in Table 2 are valid. Furthermore, process and file allocation problem is solved for each "site level" independently (step 1 of the aforementioned procedure).

Different replication policies, for example master/slave configuration, can be explored [13, 27]. Both synchronous and asynchronous replica synchronization may be supported. The replication policy for each file is defined during functional configuration (figure 4). When adopting asynchronous replica synchronization that has a relatively smaller communication cost, files are widely replicated. Thus, before analytically calculating communication cost and exhaustively solving (3) (Table 2), empirical rule-based algorithms can be applied to solve or partially solve the problem [8, 12]. 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 ones are firstly allocated). If the proposed solution is not efficient, the file allocation order is modified.

Resource allocation is performed based on decision variables. These variables, indicating the QoS imposed to physical specification, are estimated using information extracted from parameters describing elementary operations. As an example, we refer to communication cost, expressed in terms of *maximum data exchange rate (bytes/sec)* between sites. The

Table 2. Resource allocation problem representation.

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

where

- N is the overall number of sites (of the same level)
- P_c is the set of client processes
- P_s is the set of server processes
- F is the set of files
- c_{ij}^{x} represents the communication cost when accessing a replica of process or file x residing in site j from site i
- $u_{ij}^{c}, p_{ij}^{s}, f_{ij}^{f} =_{0}^{1}$, 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 to all sites. Server replicas are allocated to 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 expression (1) results in

$$\min\left(\sum_{s\in Psf} \sum_{\substack{i,j=1\\ i\neq j}}^{N} p_{ij}^{s} c_{ij}^{s} + \sum_{f\in Fs} \sum_{\substack{i,j=1\\ i\neq j}}^{N} f_{ij}^{f} c_{ij}^{f}\right) \quad (2),$$

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 expression.

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

$$\sum_{f\in Fs} \sum_{r\in Rf} \min\left(\sum_{s\in \mathrm{Pr}} \sum_{\substack{i,j=1\\ i\neq j}}^{N} p_{ij}^{sr} c_{ij}^{sr}\right) \quad (3),$$

where R_f is the set of file f replicas and P_r is the set of processes accessing each replica. It is assumed that for each process s in P_r, a replica is allocated to the same site as $r.c_{ij}^{sr}$ represents the communication cost when process s, invoked by others, accesses replica r.

estimation formula is depicted in Table 3. Communication cost c_{ij}^{s} between two sites i and j is computed based on parameters of *transfer(source site, source process, target site, target process, Amount)* elementary operation produced by decomposing all operations included in the implementation part of any component of process *s. transfer*$_{i,j}^{s}$. *Amount* denotes the *amount* parameter of *transfer* operation where *sourcesite* $= i$, source *process* $= s$ and *targetsite* $= j$. In order to estimate communication demands, *user profiles* are grouped together when they operate concurrently (as indicated by their activation parameters) in *concurrent profiles* groups. *User requests*, contained in each *user profile*, initiate the activation of multiple processes and are not concurrently invoked. They are characterized by a requested response time (*Response* parameter) that must be depicted in the communication

Table 3. Decision variable estimation example.

$c^s_{ij} = \max\left(\sum_{\text{UP in Concurrent\_Profiles}} \max_{\text{UPRinUP.User\_requests}}\left(\frac{\sum_{\text{s in processes\_invoked\_by\_UPR}}(\text{transfer}^s_{i,j}.\text{Amount}*\text{UP.Users})}{\text{UPR.Response}}\right)\right)$, where $\text{transfer}^s_{i,j}.Amount$ represents the amount of data exchanged between any process component operating in site i, invoked by the most demanding user_request of each of all concurrent user_profiles operating in the system.

cost. The summation of all data, exchanged between sites *i* and *j* involved in the invocation of process *s* by a specific user request *UPR,* multiplied by the number of users initiating this request and divided by the *UPR* requested response time provides an estimation of *data exchange rate* demand between sites *i* and *j* when process *s* is invoked by *UPR* user request.

Decision variables are estimated based on elementary operations produced by the decomposition of process component implementation parts. Since operation decomposition hierarchy facilitates detailed application functionality description, it is safe to assume that the estimated QoS demands are very accurate.

5. Case study

The proposed methodology was applied for the configuration of the integrated information system of *Greek National Diabetes Network (GNDN),* formed by the National Diabetes Institute (NDI) and 128 Medical Centres hosted in public hospitals. The information system provides 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 diabetes. Data is maintained by physicians working in diabetes medical centres and the National Diabetes Institute. Individual user groups (public, patients, researchers) access data through web services provided by *NDI portal*.

Application design and implementation was performed using Oracle product suite. Medical records are private, thus only specific clinical attributes can be publicly accessed and statistically processed. The size of medical centres differs according to the size of the hospital hosting it. There are three categories of medical centres regarding information system support, as indicating in Table 4. The required response time for all transactions is 15–20 sec, thus none of the applications is considered as time- critical. More than 1,000,000 patient records are currently supported.

Table 4. Medical centre requirements.

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

The National Diabetes Institute and medical centres are interconnected through the National Health Network, a private TCP/IP network, interconnecting public hospitals. Network connection speeds vary from 128 Kbps to 2 Mbps. In the following, we discuss in detail our experience using the proposed system specification meta-model through all configuration stages.

5.1. Functional configuration

We focus on *Medical Record* and *Statistics Provision* applications to comment on application modelling and quantitative analysis. Medical Record application is built on a "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 form invocation, field management and transaction execution. Transactions consist 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.

Since medical records are private, two different database servers were modeled (each one belonging to a different application): the *Medical DS,* maintaining medical records, and the *Informational DS,* maintaining specific fields subjected to statistical processing. It is evident that the two databases must be synchronized. *OracleApplSrv* depicts Oracle Application Server functionality and is modeled using *Web Server* model.

Two new operations were added in the operation hierarchy to integrate Oracle-specific functionality and ease *web client* description. The *form_access (form_name, no_fields, avg_fsize, processing)* depicts accessing, activating and processing of a form. This operation is decomposed into *get page, post* and *processing* operations to depict the invocation of Oracle forms as HTML pages and the filling/processing of specific form fields. The *activate_transaction(transaction_name, param_list, avg_param_size, processing)* operation depicts the activation of a transaction corresponding to an individual task, as addition of new patient, medical visit search, etc. Transactions are modelled as different components of *OracleApplSrv*. *Activate_transaction* is further decomposed into *processing, post* and *http request* operations, as depicted in figure 7. *Processing* operation indicates data processing. *Post* operation depicts passing form field values as parameter values for the transaction. The *http request* is used to invoke *OracleApplSrv* component representing the specific transaction.

In figure 7, *activate_transaction* operation is decomposed into 4 others. The number in the extended composition relationship connecting *activate_transaction* with an existing operation indicates the order of invocation. All the parameters of each invoked operation must be initiated by either a constant value or an *activate_transaction* parameter. The same operation, such as *processing*, may be invoked more than once using different initialization parameters.

Extending the operation hierarchy is important to ensure a detailed application description. If only predefined operations could be used, the same description would have to be repeatedly given (a) within the same process component implementation and (b) in different

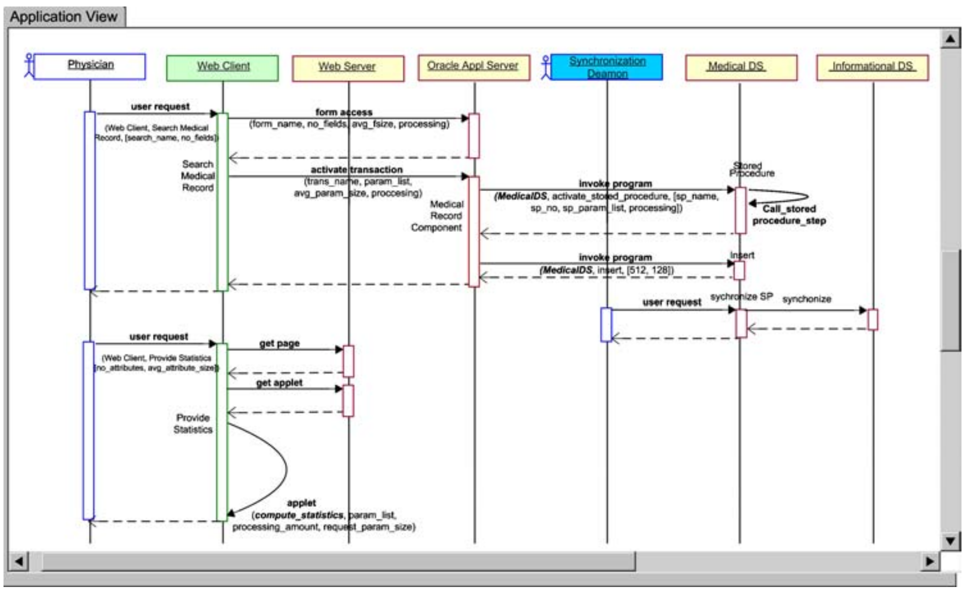

Application View
Physician
Web Client
Web Server
Oracle Appl Server
Synchronization Deamon
Medical DS
Informational DS
user request
(Web Client, Search Medical Record, [search_name, no_fields])
form access
(form_name, no_fields, avg_fsize, processing)
Search Medical Record
activate transaction
(trans_name, param_list, avg_param_size, proccesing)
Medical Record Component
invoke program
(MedicalDS, activate_stored_procedure, [sp_name, sp_no, sp_param_list, processing])
Stored Procedure
Call_stored procedure_step
invoke program
(MedicalDS, insert, [512, 128])
Insert
user request
sychronize SP
synchonize
user request
(Web Client, Provide Statistics [no_attributes, avg_attribute_size])
get page
get applet
Provide Statistics
applet
(compute_statistics, param_list, processing_amount, request_param_size)

Figure 8. Application view—medical record and statistics provision application models.

components, to represent the same functionality. Although only up to 15 operations are used to describe Medical Record application, each one of the 6 components of *Web Client* is responsible for activating a large number of elementary operations (varying from 97 to 245) corresponding to different process component implementations, such as *Web Client, OracleApplSrv*, *Medical DS*, e.t.c.

The Application View describing *Medical Record* and *Statistics Provision* applications is depicted in figure 8.

The *invoke_program(MedicalDS, stored_procedure, [sp_name, sp_no, sp_param_list, processing])* operation depicts the invocation of *MedicalDS* to execute a specific stored procedure, while an *invoke_program(MedicalDS, insert, [512, 128])* operation is included as the last operation in the *Medical Record Search* component of *OracleApplSrv* to log the specific user performing medical record search. The *call_stored_procedure_step* (*preprocessing*, *data_accessed, postprocessing)* operation is used to describe store procedure functionality. Using it, the description of stored procedures was significantly simplified. Each stored procedure consists of one to five steps. The *call_stored_procedure_step* includes the activation of *processing*, *read* and *write* operations. Since the *Informational DS* is updated only through database replication, *synchronize* component had to be implemented according to Oracle replication mechanisms. This component is invoked whenever *synchronize* operation is invoked.

Composite operations, such as *form_access*, may result in the invocation of a specific process, such as *OracleApplSrv*, more than once, even though this is not indicated in Application View. Composite operations may result in the invocation of other processes even when this is not indicated in Application View, as in the case of *applet* operation. This is due to operation decomposition, which is transparent to the system designer when defining a component implementation.

Process Statistics component invokes the *Web Server* through *get_page* and *get_applet* operations. 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 fields fulfilling predetermined constraints.

5.2. Logical configuration

Users initiate Medical Record application modules by invoking Oracle Application Server, thus a replica of Oracle Application Server and related files were placed in all medical centres for better performance. The implementation of an Oracle Database in small-sized medical centres is costly. Alternative *Medical DS* allocation scenarios were studied in order to provide the requested response time and reduce cost. After evaluating different alternatives, it was decided to place *Medical DS* replicas in all medical centres except for small ones directly connected with another one with a network connection faster than 512 Kbps. A replica of *Informational DS* was placed in NDI, since Internet access is supported through this specific site.

In functional specification, site description was performed at two levels, the GNDN level and the medical centres and NDI level. During logical configuration, site decomposition hierarchy was modified to depict the structure of the WAN network interconnecting hospitals (each hospital belongs to a regional WAN), 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 9.

A replica of *Oracle Application Server, File Server* and *Medical Database Server* is placed in the medical centre of the 3rd *Attica Regional Hospital*. Seven physicians concurrently use the Medical Record and Provide Statistics application. The corresponding user profile initiates a *web client* to access both applications. All server processes corresponding to *Provide Statistics* application are placed in *NDI* site, and thus are not depicted in this figure.

5.3. Physical configuration

Since GNDN is supported by the National Health network, the network topology was predefined.

The underlying network and database architecture could be modelled and studied using various commercial simulation tools that may measure and analyze traffic at different network layers. However, 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

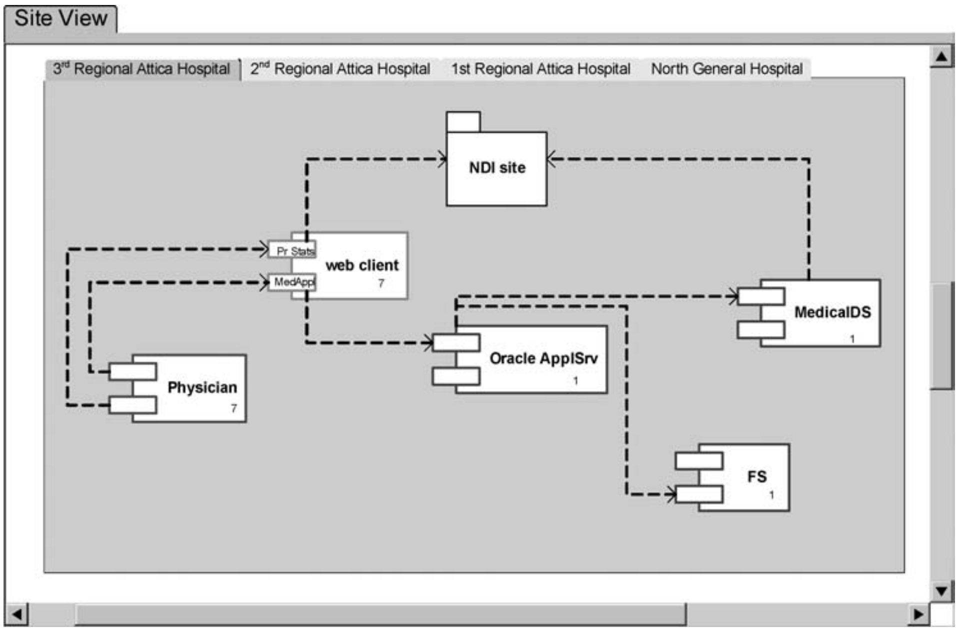


Figure 9. Site view—3rd attica regional hospital.

primitives was not feasible. The extendable *operation dictionary* concept provided a set of common high-level constructs, conforming to Oracle supported application architecture, which enable the accurate application functionality description and direct mapping of this description into QoS parameters.

6. Conclusions

We proposed a configuration methodology for web-based systems, emphasizing the symbiotic relationship between application configuration and the 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 specification) to application functionality (functional specification). The proposed model enables the exploration of dependencies between configuration stages even if these aren't obvious, since functional specification is corrected or filled, while physical specification is progressively defined.

The extendable *operation dictionary* proved to be essential for describing complex application functionality, such as the one supported by Oracle Application Server, since it enables the accurate application functionality description and the direct mapping of this description into QoS parameters that the underlying network must satisfy. This proved to be the main advantage of the proposed model. UML-like representation of model entities helped in model extension/customization, since (a) system designers are familiar with UML constructs and (b) automated code generation can be achieved.

References

1. M.F. Arlitt and C.L. Williamson, "Internet Web servers: Workload characterization and performance implications," IEEE/ACM Transactions on Networking, vol. 5, no. 5, 1997.
2. J. Cruz and K. Park, "Towards performance-driven system support for distributed computing in clustered environments," Journal of Parallel and Distributed Computing, vol. 59, no. 2, 1999.
3. A. Dutta and S. Mitra, "Integrating heuristic knowledge and optimization models for communication-network design," IEEE Transactions on Knowledge and Data Engineering, vol. 5, 1993.
4. G. Fleischanderl, G.F. Friedrich, et al., "Configuring large systems using generative constraint satisfaction," IEEE Intelligent Systems, vol. 1, 1998.
5. H. Gomaa, D. Menasce, and L. Kerschberg, "A software architectural design method for large-scale distributed information systems," Distributed System Engineering Journal, vol. 3, no. 3, 1996.
6. H. Gomaa and M. Shin, "Multiple view meta-modeling of software product lines," in Proceedings of the 8th International Conference on Engineering Complex Computer Systems, IEEE Computer Press, 2002.
7. S. Graupner, V. Kotov, and H. Trinks, "A framework for analyzing and organizing complex systems," in Proceedings of the 7th International Conference on Engineering Complex Computer Systems, IEEE Computer Press, 2001.
8. W.E. Juengst, and M. Heinrich "Using resource balancing to configure modular systems," IEEE Intelligent Systems, vol. 1, 1998.
9. P. Kaehkipuro "UML-based performance modeling framework for component-based distributed systems," Lecture Notes in Computer Science 2047, Performance Engineering, Springer-Verlag, 2001.
10. V.D. Khoroshevsky, "Modelling of large-scale distributed computer systems," in Proceedings of IMACS World Congress, Conf. 15, vol. 6, 1999.
11. A.M. Law and M.G. McComas, "Simulation software of communications networks: The state of the art," IEEE Communications Magazine, vol. 4, no. 3, 1994.
12. S.J. Lee and C.H. Wu, "A knowledged-based approach to the local-area network design problem," Applied Intelligence, vol. 4, no. 1, 1994.
13. M. Marreitti, Replication, Academic Press: London, England, 1999.
14. R. Mirandola and V. Cortellessa, "UML based performance modeling in distributed systems," Lecture Notes in Computer Science 1939, UML2000, Springer-Verlag, 2000.
15. H.L. Morgan and K.D. Levin, "Optimal program and data locations in computer networks," Communications of ACM, vol. 20, no. 5, 1977.
16. G.S. Nezlek, K.J. Hemant, and D.L. Nazareth, "An integrated approach to enterprise computing architectures," Communications of the ACM, vol. 42, no. 11, 1999.
17. M. Nikolaidou, D. Lelis, et al, "A discipline approach towards the design of distributed systems," Distributed System Engineering Journal, vol. 2, no. 2, 1995.
18. M. Nikolaidou and D. Anagnostopoulos, "A distributed system simulation modelling approach," Simulation Practice and Theory Journal, vol. 11, no. 4, 2003.
19. OMG Inc, OMG Unified Modeling Language Specification, Version 1.5, March 2001.
20. Oracle Co, "Oracle application server 10g: High availability," Oracle White Paper, January 2004.
21. S. Ramesh and H.G. Perros, "A multi-layer client-server queuing network model with non-hierarchical synchronous and asynchronous messages," Performance Evaluation, vol. 45, no. 4, 2001.
22. R. Reeser and R. Hariharan, "Analytic model of web servers in distributed environments," in Proceeding of the International Workshop on Software and Performance, Ottawa, Canada, ACM Press, 2000.
23. N.N. Savino-Vázquez et al., "Predicting the behaviour of three-tiered applications: dealing with distributed-object technology and databases," Performance Evaluation, vol. 39, no. 1-4, 2000.
24. D. Serain, Middleware, Springer-Verlag: London, Great Britain, 1999.

25. A.M. El-Shaieb, “A new algorithm for locating sources among destinations,” Operations Research, vol. 20, 1973.
26. J. Shedletsky and J. Rofrano, “Application reference designs for distributed systems,” IBM System Journal, vol. 32, no. 4, 1993.
27. M. Tan and H.J. Siegel, “A stochastic model for heterogeneous computing and its application in data relocation scheme development,” IEEE Transactions on Parallel and Distributed Computing, vol. 9, no. 11, 1998.
28. R. Willenborg, K. Brown, and G. Cuomo, “Designing websphere application server for performance: An evolutionary approach,” IBM System Journal, vol. 43, no. 2, 2004.