A Cost-oriented methodology for the design of Web based IT architectures

Danilo Ardagna
Politecnico di Milano
Via Ponzio 34/5
20133 Milano, Italy
+39 02 2399 3561
ardagna@elet.polimi.it

Chiara Francalanci
Politecnico di Milano
Via Ponzio 34/5
20133 Milano, Italy
+39 02 2399 3457
francala@elet.polimi.it

ABSTRACT
This paper proposes a design methodology of Web-based IT architectures tying organizational requirements to technical choices and costs. Information system design and optimum sizing is the result of a reconciliation of several conflicting requirements, including technical performance and costs. Web-based IT architectures involve a number of design choices with significant cost implications: the adoption of thin clients executing Web applications remotely, the choice of the number of architectural tiers over the Web, the allocation of applications on physical machines and the total number of servers involved. The main goal of this paper is the identification of a sequence of design steps, from requirements analysis to physical implementation, that allows designers to estimate the cost implications of architectural choices and, by evaluating multiple design alternatives, determine the minimum-cost architectural solution. Preliminary results from the empirical verification of the methodology indicate that for Web-based architectures cost reductions can be significant and would support the practical use of a cost-oriented approach as a complement to traditional performance evaluations.

Keywords
IT architectures, Web architectures, cost minimization.

1. INTRODUCTION
Information system design and optimum sizing is the result of a reconciliation of several conflicting requirements, including technical performance and costs, organization impact, and user acceptance. Web-based IT architectures involve a number of design choices with significant cost implications: the adoption of thin clients executing Web applications remotely, the choice of the number of architectural tiers over the Web, the allocation of applications on physical machines and the total number of servers involved. Theoretical research and practitioners often focus on specific techniques for the optimization of individual design phases, usually leading to local technical optima, with little understanding of the relationships among different choices at different design stages. This paper attempts the definition of a comprehensive design methodology. The main goal is the identification of a sequence of design steps that identify the cost implications of design choices from requirements analysis to physical implementation.

The focus is on architectural design tying the information requirements of an organization to the IT components necessary to satisfy those requirements: costs are associated with these components (cf. [12]). This process has many degrees of freedom, since different Web-based architectures can be built to satisfy a given set of requirements. The proposed methodology guides the system designer through a sequence of choices at different design steps in order to distinguish alternative solutions and evaluate their costs. The methodology helps choose the architecture that minimizes the overall cost, including hardware, software, maintenance, management and training.

The next Section reviews previous approaches and highlights the main organizational and technical variables of interest. Section 3 presents the phases of the design methodology and the corresponding architectural models. Section 4 reports and discusses experimental results from a comparative analysis of costs of different architectures and describes a prototype that implements the proposed methodology. Finally, Section 5 draws conclusions and suggests directions for future research.

2. RESEARCH BACKGROUNG AND MOTIVATION
Cost analyses of technical choices have highlighted numerous design trade-offs primarily related to the appropriate sizing and location of computing resources within an IT architecture (cf. [3]). In the early '80s a lower cost of processing capacity on smaller computers and a generally reduced software complexity have created the belief that a decentralized IT architecture would involve lower overall costs for an organization and Web-based architectures would comply with this traditional trend. A number of technical factors would instead work against decentralization. First among them, communication costs can be significantly higher in a decentralized architecture, due to a more cumbersome data retrieval and consolidation in distributed transaction processing (cf. [7]). In practice, databases were duplicated to limit remote data access through geographical networks, while raising

SAC 2002, Madrid, Spain

data consistency issues and causing additional communication load to align multiple copies of the same data.

The most general technical model answering the research question of this paper is provided by [5], studying the optimal sizing of a distributed system to minimize the costs of communication for remote transactions, processors, primary and secondary storage, while guaranteeing maximum file availability and minimum response time for queries. Francalanci and Piuri (cf. [2]) also proposed a design methodology that ties organization requirements to IT technical and management costs. They refer to the Gartner Group client-server model classification that considers applications presentation, logic and data management component allocation.

Recently, practitioners are considering re-centralization of data and application functionalities in order to decrease management effort and costs. This seems to indicate that the design of the IT architecture is often guided more by current beliefs and market offer, than by an understanding of organizational requirements and technology costs. System centralization is encouraged primarily by new thin client architectures. Thin clients execute the presentation component of applications, that is they manage the user interface, while they delegate the execution of application logic to server computers, that can be outsourced to ASPs and accessed through the Web. Outsourcing is considered an important factor that promotes thin client diffusion (cf. [4]).

Two different types of thin client architectures can be implemented, Network Computers (NC) and Windows Terminals (WT). WT are supported by a Windows NT/2000 server, while NCs can execute only pure Java applications and are usually supported by non-Windows servers, such as Sun servers. NCs manage the user interface by means of a browser which is executed locally and supports access to Web-based server applications. On the contrary, WTs typically do not locally execute a browser and require an additional server to manage WTs' access to applications that is achieved through a remote protocol (Citrix ICA or Microsoft RDP). WTs simply emulate a GUI and manage users interactions. Thin client shipments are currently growing. IDC forecast a growth of unit shipments of 90% in 2001 (in the past two years shipments growth were on average 84.7%).

WTs are the most widespread thin client platform but in the long term practitioners forecast NC shipments growth due to Java applications and Intranet/Extranet environment diffusion. Thin clients will probably replace dumb terminals but cannot replace PCs because they are suitable for the execution of data entry and office automation application, but cannot execute computation-intensive applications. Anyway ICA and RDP protocols allow remote access to windows applications by traditional PCs and new hand-held devices and practitioners suggest that the remote execution of applications when PCs are adopted can reduce the total cost of ownership (TCO). As PDAs and 3G phones will become more widespread CIOs will face the problem of planning access to enterprise servers from these new devices. Capacity planning is an important issue that CIOs have to face to guarantee system performance and availability, which are more and more critical especially for Web-based systems. With the development of n-tier systems, the design paradigm that is followed by planners can be expressed as "design large, but build small" (cf. [11]). That is, by introducing multiple tiers, additional computing capacity can be obtained by introducing new machines and sharing computation load.

Capacity planning today is achieved following two different methodologies: planners focus on system performance, which is usually measured in terms of response time, and analyze a reduced set of design alternatives since the performance analysis of a single solution is cumbersome (cf. [8]). On the other hand, practitioners refer to empirical methodologies based on costs, partly supported also by software tools such as Main Control or Gartner Group TCO Manager. However, these methodologies are not systematic and have no scientific justification/basis. This paper is an attempt to fill this literature gap by proposing a methodology that systematically and formally tackles architectural choices and their cost impact on a company's IT infrastructure.

3. DESIGN METHODOLOGY

The architectural design process ties the information system requirements of an organization to the physical components necessary to satisfy those requirements. For Web based architectures design alternatives are summarized in the next section. The following sections describe the phases of the architectural design process, the architectural model that is generated by each phase and the input required from the designer needed as an input to the final optimization algorithm.

3.1 Architectural Design Alternatives

The following design alternatives for Web-based architectures are considered:

- *Thin versus fat clients* – Thin clients manage the user interface of applications stored and executed remotely, while fat clients store and execute applications locally. The choice between fat and thin clients can significantly change the processing load on server machines.
- *Number of tiers* – The client-server paradigm can organize applications in multiple tiers. Each application tier is typically allocated on a separate machine and responds to service requests from lower tiers, while sending service requests to higher tiers.
- *Total number of servers* – The required computing capacity can be allocated on one or multiple servers, whose total number represents an architectural alternative.
- *Allocation of applications* – Different applications (or application tiers) can be allocated on separate computers and, vice versa, multiple applications can be allocated on the same computer.

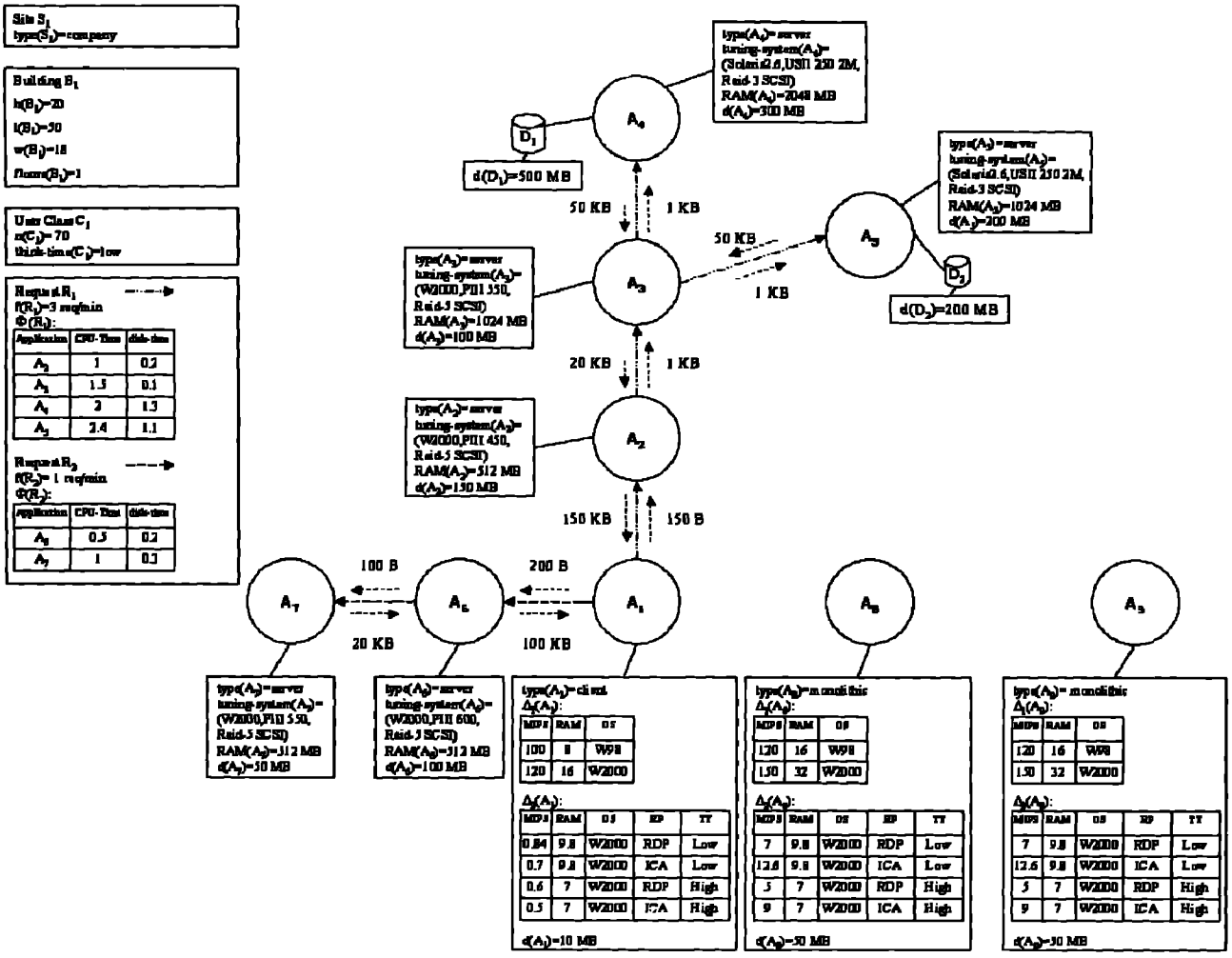

Figure 1. Graphical representation of a technology requirement model

Note that these design alternatives are mutually interdependent. For example, a higher number of tiers increases the degrees of freedom in allocating applications and improves load balancing among servers. Since this paper focuses on hardware choices, application design alternatives are not included in the list above and in the architectural models discussed in the next sections. However, the designer can define constraints on hardware alternatives imposed by previous application design choices.

The design alternatives listed above are credited an impact on the total cost of the corresponding IT architecture (cf. [10], [11]). However, the most appropriate criterion to select a solution among a set of applicable alternatives may not be cost minimization (cf. [8], [6]). First of all, technical constraints can represent a driver of choice. For example, the need to integrate new architectural components with legacy systems can constrain the selection of the hardware and network standards, such as the number of tiers or the communication protocol. Similarly, organizational constraints, such as security policies or centralization practices, may reduce the set of viable solutions. The methodology presented in this paper assumes that technical and organizational constraints translate into architectural design decisions that reduce the degrees of freedom of the cost-minimization algorithm. Constraints can be defined as predetermined solutions for one or more of the design alternatives listed above, as explained in the next sections.

3.2 Architectural Design Phases

The minimum-cost IT architecture satisfying organizational requirements and constraints is identified with three design phases. First, organizational requirements are specified by building the technology requirements model. Then, a theoretical model of the minimum-cost IT architecture satisfying organizational requirements, the infrastructural model, is built under the assumption that architectural components can meet requirements with no approximation. Theoretical components are replaced with real components in a final design phase building the minimum-cost physical model of the IT architecture. Each model of the technology architecture is translated into the model produced in the subsequent design phase by a specification activity. Specification activities involve design choices that, in turn, affect architectural costs. The mapping from a model to the following one is semi-automatic, since the designer is involved in the definition of the optimization domain and bounds. The next three sections discuss the technology requirements, infrastructural and physical models, respectively, and corresponding specification activities.

3.3 The Technology Requirements Model

Technology requirements are expressed by means of the following fundamental variables:

- Organization sites S_i, defined as sets of organizational resources (users, premises and technologies) located within a 1 Km distance from each other. Conversely, the closest resources within different sites should be separated by more than 1 Km. A 1 Km threshold distance is chosen to distinguish between different network technologies within and among sites, local and geographical, respectively.
- Buildings B_i, defined as the smallest components of an organization's premises.
- Applications A_i, defined as a set of functionalities that can be accessed by activating a single computing process. Note that commercial programs mapping into multiple processes can be represented by means of multiple applications.
- User classes C_i, defined as a group of n_i users using the same subset of applications, with common capacity requirements and operating from the same floor of a site's building.
- Requests R_i, defined as interactions among applications aimed at exchanging services according to the client-server paradigm.
- Databases D_i, defined as separate sets of data that can be independently stored, accessed and managed.

A sample technology requirements model is reported in Figure 1. As a preliminary observation, the specification of sites, buildings and user classes is critical to select and size the network components of the architecture during infrastructural design. Applications, requests and databases are instead the main drivers of infrastructural design choices related to client and server computers and to their management policies.

User classes are supposed to be located on a single floor of a building and to use the same set of applications with a common think-time. User think-time is a qualitative indicator of the frequency with which users interact with their client computer and is used in subsequent sizing activities (see Section 3.4). It has been empirically demonstrated that computing capacity requirements of client and monolithic applications depend on users' behavior. For example, users could be responsible for difficult decision-making tasks and spend part of their time in evaluating application outputs without interacting with their client computers, that is, they would have a high think-time. In this case, even if they simultaneously open numerous applications, their computing requirements would be low. It has also been found that typing speed is an accurate proxy of users' think-time (cf. [9]) and that think-time can be considered low if typing speed is above 50 hits per minute. Empirical evaluations of capacity requirements for client and monolithic applications are based on a qualitative distinction between high and low values of think-time.

Applications are classified as client, monolithic, server or external. Note that a server application can also play the role of client and convey requests to other server applications. Computing capacity requirements for client and monolithic applications are traditionally expressed in MIPS (cf. [5]). MIPS and RAM requirements are not independent of the operating system and possibly of the remote protocol adopted and, therefore, applications are associated with multiple capacity requirements for different operating systems and remote protocols. In subsequent design activities, MIPS and RAM requirements are adjusted according to users' think-time (see Section 3.4). Irrespective of their allocation on a personal computer or on a server, client and monolithic applications do not require the specification of secondary memory's performance requirements, since disks have been demonstrated not to represent a capacity bottleneck (cf. [9]).

On the contrary, server applications require the specification of disks' response time (cf. [8], [6]). Servers are modeled by means of the queuing network model. The model we adopt includes a single disk, since current secondary memory technologies based on RAID disks can be modeled as a single resource (cf. [8]). Note that DBMSs are supposed to be specified as server applications and, accordingly, databases can be more simply described through their required size of secondary memory.

Requests are initiated by a client or an external application and are answered by one or multiple server applications. If multiple server applications are involved, they coordinate by triggering one another in a sequence and building the response incrementally according to the client-server paradigm. The first one is in charge of conveying the response to the requesting application.

Server applications are characterized by the CPU and disk demanding times required to support the execution of requests. Demanding times are supposed to be evaluated on a tuning system. This allows the estimate of demanding times on a different system by means of benchmarking data (cf. [8]). Tuning systems are associated with applications as opposed to individual requests. Demanding times are in fact evaluated on a single system for all the requests served by an application.

3.4 The Infrastructural Design Model

The goal of the infrastructural design phase is to build a virtual IT architecture satisfying technology requirements under the assumption that architectural components can meet requirements with no approximation. Since physical machines have a discrete distribution in the requirements space, during physical design virtual architectural components will be associated with real equipment. A Technology Requirements Model can be associated with N Infrastructural Models obtained through the exploration of the solution domain defined by the designer. On the contrary, each Infrastructural Model can be assigned a single Physical Model satisfying infrastructural requirements and minimizing costs. The solution domain is defined specifying the following architectural design choices:

1. Type of client computers – Client computers can be constrained to fat, thin, fat hybrid or terminal (host-based) solutions.
2. Number of computing levels of requests – The number of computers involved in responding to requests can be constrained to a specific value.

3.4.1 Number of computing levels of requests

As discussed in Section 3.3, requests can involve multiple server applications triggering one another sequentially. Server applications can be executed by the same or by different computers. If separate computers execute server applications, the resulting hardware architecture is referred to as multi-level and the number of levels evaluates to the maximum number of distinct computers involved by the same request. For each request, the upper bound of the number of levels of the corresponding

Table 1 – Summary of virtual computing resources and sizing

Virtual Computing Resource	Symbol	Variables	Analytical Formulation	Notes
Virtual Server	VS_i	Frequency of requests	$f(R_i)$	
		Primary Memory	$RAM = \sum_{A_i \in VS_i} RAM(A_i)$	
Virtual application server for WT/HFC	VAS_i	Computing Power	$MIPS = n(C_i) \cdot p(C_i) \cdot \max_{A_i \in VAS_i}(MIPS(A_i, OS, RP, think-time(C_i)))$	MIPS(A_i, OS, RP,think-time(C_i)) returns MIPS required to support execution of application A_i on the target operating system OS under the specified remote protocol RP for a user of specified think-time; $p(C_i)$ returns the percentage of concurrent users in class C_i
		Primary Memory	$RAM = n(C_i) \cdot p(C_i) \cdot \sum_{A_i \in VAS_i} RAM(A_i, OS, RP, think-time(C_i))$	RAM(A_i, OS, RP) returns RAM required to support execution of application A_i on the target operating system OS under the specified remote protocol RP for a user of specified think-time.
Virtual fat client	VFC_i	Computing Power	$MIPS = \max_{A_i \in VFC_i}(MIPS(A_i, OS))$	MIPS(A_i, OS) returns MIPS required to support execution of application A_i on the target operating system OS
		Primary Memory	$RAM = \sum_{A_i \in VFC_i} RAM(A_i, OS)$	RAM(A_i, OS) returns RAM required to support execution of application A_i on the target operating system OS
Virtual HFC	$VHFC_i$	Computing Power	$MIPS = \max_{A_i \in VHFC_i}(MIPS(A_i, OS))$	MIPS(A_i, OS) returns MIPS required to support execution of application A_i on the target operating system OS
		Primary Memory	$RAM = \sum_{A_i \in VHFC_i} RAM(A_i, OS)$	RAM(A_i, OS) returns RAM required to support execution of application A_i on the target operating system OS

architecture is the number of applications involved; the lower bound is always two, corresponding to the execution of all server applications on the same computer.

At this stage the designer has to specify for each request path the acceptable numbers of levels and sets of applications that have to be executed by the same server. For the same path multiple number of levels and allocations can be defined. For example, for the same path the 3 and 4 level alternative may both be explored.

3.4.2 The infrastructural design process

Virtual computing resources are associated with server applications and user classes and sized by accounting for computing requirements of all user classes, as specified in Table 1. Virtual computing resources represent hardware components that meet requirements and satisfy constraints with no approximation. The following virtual computing resources are considered during infrastructural design:

1. *Virtual servers* – Virtual servers represent computers that execute at least one server application for one user class. They are described by their primary and secondary memory and by the frequency of requests to be served. The frequency of requests is used to evaluate the capacity of servers during physical design.
2. *Virtual WT and HFC servers* – Virtual WT and HFC servers represent computers that execute at least one client or monolithic application for one user class constrained to use a WT or HFC client architecture. They are described by their computing capacity, primary and secondary memory.
3. *Virtual fat clients and HFC* – Virtual fat clients and HFC both represent computers that execute at least one client or monolithic application. They are described by their computing capacity, primary and secondary memory.
4. *Virtual thin clients* – Virtual thin clients represent computers that execute or emulate the presentation component of an application. Different from other architectural components, they are not described quantitatively, but simply categorized as either low or high performance devices.

Table 1 reports the formal description of virtual computing resources and summarizes their sizing rules. The following criteria are applied to associate virtual computing resources with server applications and user classes:

1. Each computing tier defined in Section 3.4.1 is assigned a virtual server.
2. Each user class C_i that has been constrained to a fat client architecture is assigned $n(C_i)$ virtual fat clients. Each fat client is assumed to execute all client and monolithic applications in $\{(A_j,C_i)\}$ (this set defines the applications used by class C_i).
3. Each user class C_i that has been constrained to a NC thin client architecture is assigned $n(C_i)$ virtual network computers.
4. Each class C_i that has been constrained to a WT thin client architecture is assigned $n(C_i)$ virtual windows terminals and a

WT virtual server. The WT virtual server is assumed to execute all client and monolithic applications in $\{(A_j,C_i)\}$.

5. Each class C_i that has been constrained to a hybrid fat client architecture is assigned $n(C_i)$ HFC virtual clients and a HFC virtual server.

Secondary memory is sized as the summation of secondary memory requirements $d(A_i)$ of applications A_i executed by the virtual computer and of secondary memory requirements $d(D_j)$ of databases D_j in $\{(D_j,A_i)\}$ that are managed by applications A_i. Similarly, primary memory evaluates to the summation of RAM requirements of the applications A_i simultaneously executed by the virtual computer, of $RAM(A_i)$ for server applications and of the RAM values specified in Δ_1 and Δ_2 for client and monolithic applications (see Figure 1). The computing capacity of PCs is obtained as the maximum value of MIPS required by applications that are executed locally. In the same way, capacity requirements for Virtual WT and HFC servers are evaluated by considering the maximum value for MIPS required by applications that are remotely executed multiplied by the number of concurrent users.

3.5 The Physical Design Model

Physical design transforms virtual computing resources into commercial components according to a cost-minimization process. The overall output of the physical design phase is the minimum-cost IT architecture satisfying technology requirements and constraints. As noted before, multiple infrastructural models can be built in compliance with technology requirements. The number of infrastructural models to be compared can be high and cost minimization can represent a computationally complex process. The algorithmic approach to cost minimization adopted in the prototype tool implementing the methodology is described in Section 4. As a general consideration, optimization iterates the following two steps in order to design the minimum-cost architecture:

1. selection of an infrastructural model;
2. association of commercial components with virtual computing resources of the selected infrastructural model.

Commercial components are selected for association with virtual resources as the lowest-cost devices satisfying the following criteria:

- *Virtual servers*: the utilization of physical resources CPU and disk should be lower than 60%, in order to guarantee an acceptable response time (cf. [8]). With values of utilization greater than 60%, small variations of throughput would cause a substantial growth of response time and, overall, performance would become unreliable.
- *Virtual WT and HFC servers, virtual fat clients and HFC*: computing capacity is greater than that of corresponding virtual resources.
- The primary and secondary memory of commercial components should be greater than those of corresponding virtual resources.
- Virtual WT: design criteria that discriminate low and high performance devices are satisfied.

4. EMPIRICAL VERIFICATIONS

The methodology has been empirically validated by verifying that the design alternatives listed in Section 3.1 can be optimized with significant cost advantages. This validation has been supported by a prototype software tool, ISIDE (*Information System Integrated Design Environment*), that implements the methodological design steps described in Section 3. ISIDE's optimizer is based on the *tabu search* heuristic algorithm (cf. Aarts and Lenstra 1997).

In the following we report the cost reductions that can be obtained by applying the cost-minimization methodology to the design alternatives listed in Section 3.1.

- *Thin versus fat clients* – To evaluate cost reductions from the optimization of the thin vs. fat client alternative, the data entry worker (DEW), structured task worker (STW) and knowledge worker (KW) user classes have been considered, as described in [9]. DEW, STW and KW have been assigned PCs and WTs of low, intermediate and high profile, respectively. The number of users in each class has been increased from 5 to 2500. A system life time of 3 years has been considered. In the first year, acquisition and management cost have been accounted for, while for next two years only management cost have been included. Management costs for WTs and PCs have been estimated as described in [10]. In particular, in [10] it is assumed that management costs for WTs are between 25% and 35% lower than management costs for PCs. Analyses have been performed for the extreme values of this cost range. Results show that, on average, the WT solution involves a reduction of TCO of 19%, 23% and 30% for DEW, STW and KW, respectively, when WTs' management costs are hypothesized to be 25% lower than PCs' management costs. Reductions grow to 35%, 39% and 44% when WTs' management costs are hypothesized to be 35% lower.
- *Number of tiers* – A request typology involving 4 server processes has been considered to evaluate the 2, 3, 4 and 5 tier alternatives. RAM requirements for the four processes were 256, 256, 512 and 512 MB. Two different scenarios have been considered. In the first scenario, the application load was uniformly distributed among computing tiers and the frequency of requests varied from 0 to 7.54 req/min (the frequency increase step was 0.01). In the second scenario, a one second demanding time has been considered for the fourth and fifth tiers and frequency varied from 0 to 3.54 req/min. A disk demanding time of 0.1 sec. on a 5 disk Raid-5 mono-channel Ultra SCSI disk system has been considered. The average cost reduction from optimization was 46.44% and 33.04% for the first and second scenario, respectively.
- *Total number of servers* – Two different server farm analyses have been performed. The first referred to a server application that generated requests which required 0.5 sec. of CPU in a tuning system based on a PIII 550. Request frequency varied from 0 to 37.68 req/min. (step 0.01). RAM requirements were 512 MB and disk demanding time was 0.1 sec. on a 5 disk, Raid-5 mono-channel Ultra SCSI disk system. The optimization process delivered a 79.92% average cost reduction. A server farm supporting WT has also been considered. Analyses were performed for DEW, KW and STW users. The number of users varied from 5 to 2100 users with a 5-user increase step. The optimization process delivered an average cost reduction of

45.39%, 29.75% and 38.40%, respectively. In this analysis, management costs were excluded, since recent software management tools allow the administration of a server farm with an effort comparable with a single server.

- Server sharing – Two server applications have been considered and a single-server has been compared with a two-server architecture. We supposed that applications require 512 MB of RAM and support two types of requests. Three different computing loads have been considered:
  - Both types of requests required 0.5 sec. of CPU on a PIII 550 tuning system and frequencies varied from 0 to 3.56 req/min.
  - The first type of request required 0.5 sec. of CPU and the second 1 sec. on a PIII 550 and frequencies varied from 0 to 3.56 and 1.78 req/min respectively.
  - The first type of request still required 0.5 sec. of CPU and the second 1 sec. on a PIII 550, but frequencies varied from 0 to 1.87 and 2.82 req/min, respectively.

  Disk demanding time was 0.1 sec. on 5 disk, Raid-5 mono-channel Ultra SCSI disk system in all cases. The average cost reduction was 50.03%, 54.67% and 37.80%, respectively.

  Then, two user classes have been considered and the single server vs. two server option was evaluated. The analysis was performed for DEW, KW and STW, with the following distribution of users across classes:
  - DEW: both users classes varied from 5 to 1090 users, from 5 to 545 and 1635 and from 5 to 270 and 1900. Cost reductions were 11.08%, 12.88% and 13.33%, respectively.
  - KW: both users classes varied from 5 to 300 users, from 5 to 165 and 495 and from 5 to 80 and 575. Cost reductions were 26.35%, 21.65% and 22.79%, respectively.
  - STW: both users classes varied from 5 to 235 users, from 5 to 115 and 350 and from 5 to 60 and 410. Cost reductions were 30.63%, 24.01% and 24.04%, respectively.

  Either the single server or the two server option was preferred depending on the total number of users.

5. CONCLUSIONS

The proposed methodology allows the identification of the architectural solution that minimizes costs, against different information system requirements and multiple design alternatives. Preliminary results from the empirical verification of the methodology indicate that cost reductions can be significant and would support the practical use of a cost-oriented approach as a complement to traditional performance evaluations. On the contrary, experience-driven decisions can lead to sub-optimal architectural choices, since analyses have shown that the minimum-cost solution is often different from the expected solution based on the professional literature. In turn, this would encourage future research to both extend the methodology and improve the support tool towards its practical application. In particular, future work will consider network design alternatives and will include the analysis of legacy systems and of their impact on the optimization process.

6. REFERENCES

[1] Aarts, E., Lenstra, J. K. Local Search in Combinatorial Optimization. John Wiley & Sons Ltd, 1997.

[2] Francalanci, C., Piuri, V. Designing information technology architectures: a cost-oriented methodology. Journal of Information Tecnology, 1999.

[3] Guengerich, S. Downsizing Information Systems. Sams Publishing, 1992.

[4] Horowitz, A.,S. Thin clients tapped in times of change. Mergers and acquisition, growth of ASPs push technology further into the mainstream. Information Week www.informationweek.com.

[5] Jain, H. K. A comprehensive model for the design of distributed computer systems. IEEE transactions on software engineering. 13(10), 1092-1104, 1987.

[6] Lazowska, E. D., Zahorjan, J., Graham, G. S., Kenneth, C. S. Quantitative System Performance Computer system analysis using queueing network models. Prentice-Hall, 1984.

[7] Lee, H., Shi, Y., Stolen, J. Allocating Data Files over a Wide Area Network: Goal Setting and Compromise Design. Information & Management, 26, 85-93, 1994.

[8] Menascé, D. A., Almeida, V. A. F. Scaling for E-business. Technologies, models, performance and capacity planning. Prentice-Hall, 2000.

[9] Microsoft. Windows 2000 Terminal Services Capacity and Scaling. www.microsoft.com/windows2000/library/technologies/terminal/tscaling.asp.

[10] Molta, D. Thin Client computers come of age. Network Computing www.networkcomputing.com/1009/1009buyers1.html.

[11] Scheier, R.,L. Scaling up for e-commerce. Computerworld www.computerworld.com.

[12] Zachman, J. A. A framework for information system architecture. IBM System Journal. 38(2), 454-470, 1999.