2003

A Multi Model Algorithm for the Cost Oriented Design of the Information Technology Infastructure

Danilo Ardagna
Dipartimento di Elettronica e Informazione Politecnico di Milano, ardagna@elet.polimi.it

Chiara Francalanci
Dipartimento di Elettronica e Informazione Politecnico di Milano, francala@elet.polimi.it

Marco Trubian
Dipartimento di Elettronica e Informazione Politecnico di Milano, trubian@dsi.unimi.it

A Multi-Model Algorithm for the Cost-Oriented Design of the Information Technology Infrastructure

Danilo Ardagna
Dipartimento di Elettronica e Informazione
Politecnico di Milano
20133 Milan, Italy
Phone:+39 02 2399 3561, Fax:+39 02 2399 3411
ardagna@elet.polimi.it

Chiara Francalanci
Dipartimento di Elettronica e Informazione
Politecnico di Milano
20133 Milan, Italy
Phone:+39 02 2399 3457, Fax:+39 02 2399 3411
francala@elet.polimi.it

Marco Trubian
Dipartimento di Scienze dell'Informazione
Università degli studi di Milano
20135 Milan, Italy
Phone:+39 02 503 16230, Fax:+39 02 503 16253
trubian@dsi.unimi.it

Abstract

Multiple combinations of hardware and network components can be selected to design an information technology (IT) infrastructure that satisfies performance requirements. The professional criterion to deal with these degrees of freedom is cost minimization. However, a scientific approach has been rarely applied to cost minimization and a rigorous methodological support to cost issues of infrastructural design is still lacking.

The methodological contribution of this paper is the representation of complex infrastructural design issues as a set of four intertwined cost-minimization sub-problems: two set-coverings, a set-packing and a min k-cut with a non linear objective function. Optimization is accomplished by sequentially solving all sub-problems with a heuristic approach and finally tuning the solution with a local-search approach.

The methodology is empirically verified with a software tool including a database of costs that has also been built as part of this research. The work shows how an overall cost-minimization approach can provide significant savings and indicates how corresponding infrastructural design rules can substantially differ from the local optima previously identified by the professional literature.

Keywords

Distributed systems, information technology infrastructure, cost minimization, design tool.

1. Introduction

The information technology (IT) infrastructure is comprised of the hardware and network components of an information system (Menascé and Almeida 2000). In infrastructural design of information systems, different combinations of hardware and network components can satisfy given performance requirements and, accordingly, cost minimization has multiple degrees of freedom. The literature distinguishes between two macro design alternatives, related to the selection of hardware and network components, respectively. The first alternative is how to distribute the overall computing load of a system onto multiple machines (Gavish and Pirkul 1986; Jain 1987; Aue and Brew 1994). The second is where to locate machines that need to exchange information in order to minimize network costs (Mohapatra 1998; Lui and Chan 2002). Design decisions on both alternatives are strongly inter-related. Intuitively, different allocations of computing load can change the communication patterns among machines and modify the economics of corresponding network structures. From the inter-relations between macro design alternatives, the cost-minizing design of an IT infrastructure raises a NP-hard optimization problem.

Cost analyses have been primarily addressed by the professional literature, which evaluates selected infrastructural choices to provide cost benchmarks and practical design rules (Gartner Group 2002, Main Control 2002). In contrast, a scientific approach has been rarely applied to cost analyses and a rigorous methodological support to cost minimization is still lacking. A likely reason for this lack of attention is the empirical nature of costs whose value as a function of design choices can only be obtained through empirical measure in real computer systems (Parker et al. 1988; Willcocks 1992). This makes the scientific verification of methodological approaches particularly cumbersome, as it requires empirical data.

This paper proposes a methodology that supports the selection of a combination of hardware and network components that minimize costs. Infrastructural design alternatives are organized within a methodological framework and are provided a formal representation suitable for optimization. Then, computational complexity is tackled through a sound decomposition of the overall NP-hard optimization problem into sub-problems that can be solved with operations research techniques. Four intertwined cost-minimization sub-problems are identified: two set-coverings, a set-packing and a min k-cut with a non linear objective function. Optimization is accomplished by sequentially solving all sub-problems with a heuristic approach and finally tuning the solution with a local-search approach. The methodology is empirically verified with a database of costs that has also been built as part of this research. The paper shows how an overall cost-minimization approach can provide significant savings and indicates how optimization results can substantially differ from the local optima previously identified by the professional literature.

The next Section reviews previous approaches and highlights the main organizational and technical variables of interest. Section 3 presents the design methodology and optimization process. Section 4 discusses the experimental results of cost analyses for different infrastructural design choices. Conclusions are drawn in Section 5.

2. Research Background and Motivation

The historical cost-minimizing design principle was *centralization*, which was advocated to take advantage of hardware scale economies according to Grosch's law (Grosch 1959). A further attempt to formulate a general design principle has been made in the mid '80s, when Grosch's law has been revised as "It is most cost effective to accomplish any task on the least powerful type of computer capable of performing it" (Ein-Dor 1985). *Decentralization* and its operating rule, referred to as *downsizing*, became the methodological imperative for cost-oriented infrastructural design. The corresponding infrastructural design guideline, which has been effectively summarized as "think big, but build small", is still considered valid by practitioners (Scheier 2001).

Although academic studies have challenged the generality of the decentralization principle (Gavish and Pirkul 1986; Jain 1987), the empirical recognition of the cost disadvantages of decentralization has only occurred in the '90s with the observation of client-server costs. From an infrastructural perspective, client-server can be seen as an application paradigm that allows personal computers (PCs) to share computing load with mainframes, thus facilitating their replacement with cheaper mini-computers in compliance with the downsizing principle. However, the expected reductions of infrastructural costs have not been verified. Addressing this failure, empirical studies have showed that initial acquisition expenses represent at most 20% of the total cost of a computer over its life cycle (Willcocks 1992). As a consequence, the minimization of acquisition costs does not deliver the most convenient infrastructural design solutions. The concept of "total cost of ownership" (TCO) has been introduced and defined as the summation of both investments and management costs of infrastructural components (Faye Borthick and Roth 1994). It has been observed that while decentralization reduces investment costs, it increases management costs, due to a more cumbersome administration of a greater number of infrastructural components (Moad 1994; Stevens 1997).

Recentralization has been thereafter considered to reduce management costs. The rationale for recentralization is that the client-server paradigm can be extended to allow multiple machines to share computing load (Dewire 1997). Applications can be designed to be split into multiple modules, called *tiers*, each of which can be allocated on a different machine (Menascé and Almeida 2000). Multi-tier applications give rise to multi-tier infrastructures, that offer greater flexibility to implement the most convenient load sharing among multiple machines.

Thin clients (TCs) are currently proposed as a less expensive alternative to personal computers that can be exploited through a recentralization initiative (see Table 1, Molta 1999b). Thin clients have lower computing capacity than PCs, which is sufficient for the execution or the emulation of the presentation tier, but requires the recentralization of the application logic on a server. It has been empirically verified that thin clients have management costs 20-35% lower than personal computers (Molta 1999a; The Tolly Group 1999. Furthermore, the *Independent Computing Architecture* (ICA) and *Remote Desktop Protocol* (RDP) standards allow remote access to the application logic by traditional PCs. This translates into hybrid configurations of PCs which execute only a subset of client and monolithic applications which will be referred to in the following as *hybrid fat clients* (HFCs).

An application tier can also be simultaneously allocated on multiple coordinated machines, known as *server farm* (Harchol-Balter and Downey 1997; Menascé and Almeida 2000). Each

computer within a server farm autonomously responds to a subset of service requests addressed to the application tier, thus sharing the overall computing load with other computers within the same farm. This load sharing allows downsizing and reduces acquisition costs. Furthermore, it has limited negative effects on management costs, since server farms are equipped with software tools that allow the remote management and simultaneous administration of all servers (Microsoft 2000b). Server farms help limit the cost downside of recentralization and, overall, contribute to shift cost trade-offs towards centralization.

Another important concern in infrastructural design is the reuse of existing components, referred to in the following as *legacy systems.* Legacy systems have often a high residual economic value and, thus, their reuse becomes a relevant choice in infrastructural design and can shift cost-trade-offs (Bisbal et al. 1999). Furthermore, they can be upgraded and their life cycle can be extended over a significantly longer period of time with limited additional investments. Even if current professional guidelines generally recommend recentralization, the reuse of legacies may induce different design choices, reinforcing the need for a rigorous optimization approach.

Table 1 summarizes the infrastructural design alternatives that generate centralization-decentralization cost trade-offs. Overall, current design rules generally encourage solutions to these design alternatives that translate into a recentralization of hardware components. However, most research efforts addressing centralization-decentralization issues lack scientific rigor and only a few academic studies have attempted a more systematic analysis of cost issues in infrastructural design (Gavish and Pirkul 1986; Jain 1987). The goal of this paper is to support the cost-oriented design of modern IT infrastructures with a rigorous optimization approach to help the scientific verification of empirical design rules.

Macro-alternative	Sub-alternative	Description
How to distribute the overall computing load of a system onto multiple machines.	Client typology, thin vs. fat vs. hybrid fat client (HFC)	Thin clients manage the user interface of applications stored and executed remotely, while fat clients store and execute applications locally. Hybrid fat clients (HFCs) behave both as fat and thin clients depending on the specific application.
	Number of tiers	The client-server paradigm organizes applications in multiple tiers. Each application tier can be allocated on a separate machine and responds to service requests from lower tiers.
	Total number of servers	The required computing capacity can be allocated on one or multiple servers, organized as server farms, 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.
	Reuse of legacy systems	Requirements can be satisfied by means of either new or legacy systems. Legacy computers can also be upgraded to satisfy increasing capacity requirements.
Where to locate machines that need to exchange information.	Location of servers	Servers can be located in different sites, although all servers within the same server farm must be located in the same site.
	Network topology and standards	Sites can be connected with different routing policies and through different logical and physical communication standards.

Table 1 – Infrastructural design alternatives that generate cost trade-offs.

3. Design Methodology and Optimization Algorithm

From a methodological standpoint, revisiting centralization-decentralization trade-offs requires the representation of design alternatives in Table 1 as a single cost-minimization problem. Note, that design alternatives in Table 1 are not always applicable and can be constrained by drivers of choice different from cost minimization (Lazowska et al. 1984; Menascé and Almeida 2000). The methodology presented in this paper assumes that constraints can be defined as predetermined solutions for all design alternatives listed above, as explained in the next Sections.

In this first version of the methodology, the last design alternative in Table 1 is pre-constrained to a specific topology and standard for both local and geographical networks. LANs that connect different buildings within the same site are constrained to the extended-star topology. Different sites are constrained to be connected through an IP-based Virtual Private Network (VPN). In this way, network design is not explicitly addressed. However, the methodology includes both a sizing and a costing step for network components. This provides a necessary input for the evaluation of total infrastructural costs and allows preliminary analyses of the impact of network costs on hardware design choices.

The goal of the methodology is to select a combination of infrastructural components that minimizes costs while satisfying requirements. This involves an initial specification of *technology requirements*, which will be described in the next Section and a *cost-minimization process*, which is then presented in Section 3.2.

3.1 Technology requirements

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 (and connected through a LAN).
- *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.
- *User classes C_i*, defined as a group of n_i users using the same subset of applications, with common capacity requirements.
- *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.

The specification of sites, buildings and user classes is critical to select network components during optimization. Applications, requests and databases are the main drivers of design choices related to client and server computers.

Technology requirements are specified by modeling the characteristics of requirement variables, summarized in Table 2, and by describing their mutual relationships, as shown in Table 3. Legacy components that should be reused are specified as shown in Table 4.

Requirement variable	Symbol	Characteristics
Organization site	S_i	*type(S_i)*: it indicates whether S_i represents a *company*'s or an *external* site.
Building	B_i	*floors(B_i)*: total number of floors of building B_i.
Class of users	C_i	*n(C_i)*: number of users in class C_i. *think-time(C_i)*: average think time of users in class C_i, which can be either *high* or *low*. *p(C_i):* average percentage of concurrent users in class C_i that execute client or monolithic applications.
Application	A_i	*type(A_i)*: it indicates whether A_i represents a *client*, a *monolithic*, a *server* or an *external* application. *d(A_i)*: required size of secondary memory. If *type(A_i)=client Ú monolithic* the following characteristics should be specified: *MIPS(A_i,OS)*: Computing capacity needed to support the execution of application A_i on a client computer with operating system OS. *RAM(A_i,OS)*: Primary memory needed to support the execution of application A_i on a client computer with operating system OS. *MIPS(A_i, OS, RP, TT)*: Computing capacity needed to support the execution of application A_i for a single user with think time TT, on a server computer with operating system OS and remote protocol RP. *RAM(A_i, OS, RP, TT)*: Primary memory needed to support the execution of application A_i for a single user with think time TT, on a server computer with operating system OS and remote protocol RP. If *type(A_i)=server* the following characteristics should be specified: *tuning-system(A_i)=*(OS, processor, disk): tuple of operating system OS, processor, and disk technology of the reference tuning system for application A_i. *RAM(A_i)*: primary memory required to support the execution of application A_i.
Request	R_i	*f(R_i)*: average frequency of request R_i for a single user or external application. *Server(R_i)= {A_j}*: set of server applications involved in the execution of R_i. *Request-data(R_i,A_j,A_k)*: data exchanged from the triggering A_j and responding application A_k to support the execution of R_i. *Response-data(R_i,A_j,A_k)*: data exchange from the responding application A_k and triggering application A_j to support the execution of R_i. *CPU-time(R_i,A_j)*: CPU demanding time of server application A_j to support the execution of R_i on A_j's tuning system. *Disk-time(R_i,A_j)*: Disk demanding time of server application A_j to support the execution of R_i on A_j's tuning system.
Database	D_i	*d(D_i)*: required size of secondary memory.

Table 2 – Requirement variables and corresponding characteristics.

Relationship among requirement variables	Symbol	Characteristics
Location of buildings	*{(S$_i$,B$_j$)}*	Set of pairs (S$_i$,B$_j$) indicating that building B$_j$ is located in site S$_i$.
Location of user classes	*{(C$_i$,B$_j$,f)}*	Set of tuples (C$_i$,B$_j$,f) indicating that user class C$_i$ is located in building B$_j$ at floor f.
Location of external applications	*{(S$_i$,A$_j$)/ type(S$_i$)=ext ernal $\tilde{U}$ type(A$_j$)=ext ernal}*	Set of pairs (S$_i$,A$_j$) indicating that application A$_j$ is located in site S$_i$ where both S$_i$ and A$_j$ are external.
Use of applications	***Q****={(A$_i$,C$_j$)}*	Set of pairs (A$_i$,C$_j$) indicating that application A$_i$ is used by user class C$_j$.
Concurrency among users of server applications	*{p(A$_i$,C$_j$)/typ e(A$_i$)= server}*	Set of average percentages of concurrent users of server application A$_i$ in user class C$_j$.
Data management	***D****={(D$_i$,A$_j$)}*	Set of pairs (D$_i$,A$_j$) indicating that database D$_i$ is managed by application A$_j$.

Table 3 – Relationships among requirement variables.

Applications are classified as *client*, *monolithic*, *server* or *external*. Note that DBMSs are supposed to be specified as server applications and, accordingly, databases are simply described by the size of secondary memory that they require (database are stored in the physical server that support DBMS execution). Also note that application tiers abide by the same technical definition as applications. It is hypothesized that application tiers are specified as a set of applications {A$_i$} exchanging requests and characterized by the same operating system.

Legacy component	Symbol	Configuration	Description
Legacy server	LS_i	*Configuration= (OS, processor, disk technology, disk size, RAM)*	Tuple of operating system OS, processor, disk technology, disk size and RAM installed on legacy server LS_i.
		Upgrade={(OS, processor, disk technology, disk size, RAM, cost)}	Set of tuples describing upgraded configurations of a legacy server.
		ResidualValue	Residual economic value of a legacy server.
Legacy PC	LPC_i	*Configuration=(OS, processor, disk technology, disk size, RAM)*	Tuple of operating system OS, processor, disk technology, disk size and RAM installed on legacy PC LPC_i.
		Upgrade={(OS, processor, disk technology, disk size, RAM, cost)}	Set of tuples describing upgraded configurations of a legacy PC.
		ResidualValue	Residual economic value of a legacy PC.
Legacy thin client	LTC_i	*Configuration= (processor, RAM)*	Tuple of processor and RAM installed on legacy thin client LTC_i.
		Upgrade= {(processor, RAM, cost)}	Set of tuples describing upgraded configurations of a legacy thin client.
		ResidualValue	Residual economic value of a legacy thin client.

Table 4 – Legacy components.

3.2 Cost-minimization algorithm

The optimization problem has been split into four intertwined sub-problems, which correspond to well-structured problems of the operations research literature (see Figure 1). A final overall re-optimization step that implements a tabu-search approach is also introduced, in order to improve the local optimum that is found through the isolated solution of the four sub-problems. The following sub-problems have been identified:

1) *Client optimization*: user classes are assigned to minimum-cost client computers that satisfy constraints.

2) *Server optimization*: server applications are assigned to minimum-cost machines that satisfy computing requirements and constraints (see Section 3.3.4).

3) *Server localization*: server machines identified by solving sub-problems (1) and (2) are allocated to sites by minimizing overall network and management costs (see Section 3.4.5).

4) *Reuse of legacy systems*: server machines identified by solving sub-problems (1) and (2) and assigned to organization sites by solving sub-problem (3) are replaced with legacy machines to further reduce acquisition costs (see Section 3.4.6).

Note that physical components, either legacy or new are selected as the lowest-cost devices satisfying requirements. A complete specification of sizing rules applied to select physical components that satisfy requirements is provided in (Ardagna and Francalanci 2002). The formalization of optimization sub-problems can also be found in (Ardagna et al. 2002). A brief discussion of the four sub-problems is provided in the following.
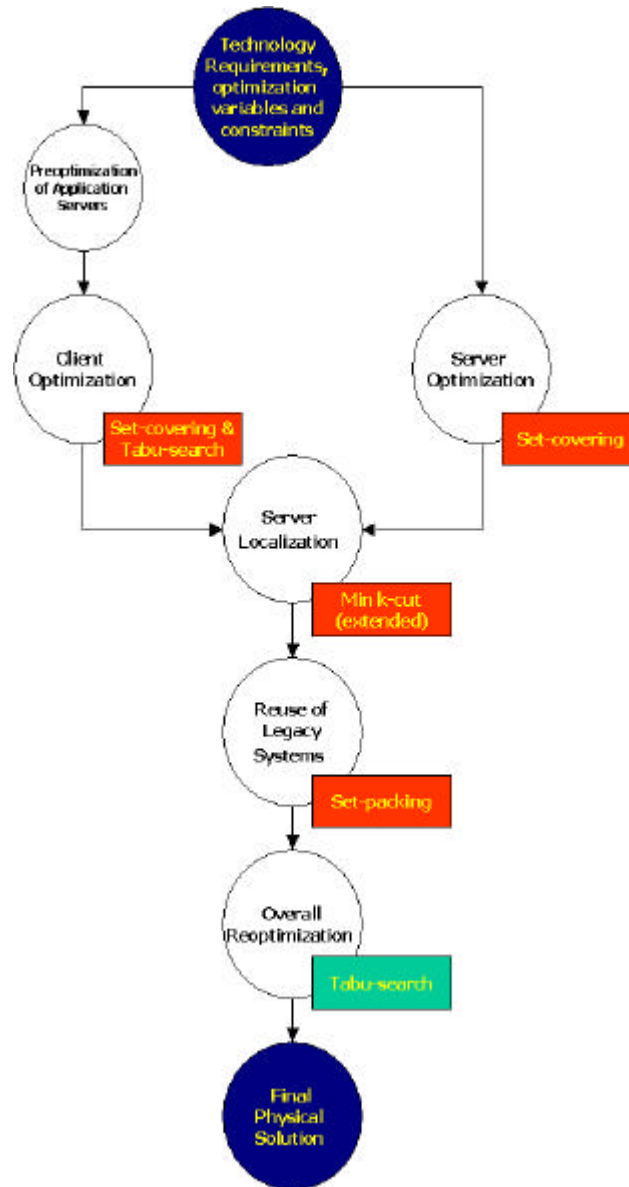
Figure 1 – Optimization steps of the multi-model algorithm.

3.2.1 Client optimization

In this phase, decisions are made on (a) which client computer is assigned to each user class, (b) which servers are necessary, (c) how servers are connected to thin clients and HFCs, and (d) where servers are located.

Pre-optimization

For each couple of values of RAM and MIPS corresponding to capacity requirements of a server (Ardagna and Francalanci 2002), an off-line exhaustive search has been implemented to determine the value v(RAM, MIPS) of the minimum cost sever(s) configuration which can satisfy memory and computing capacity requirements.

Optimization process

This optimization phase has been split into three steps.

In the first step each user class C_i is assigned to a set of $n(C_i)$ identical client computers. All machines are located in the site $S(C_i)$, where C_i resides.

In the second step the solution obtained in the first step is improved by connecting sets of client computers to the same server. The assignment of TCs and HFCs to servers is modeled as a *Set Covering Problem* (SCP) (Papadimitriou and Steiglitz 1982; Ardagna et al. 2002).

In the third step a local search based algorithm attempts an improvement of the solution provided by the first two steps. Previous choices are modified by assigning a user class to a different type of client computer. Hence, the neighborhood of a solution is defined by all solutions that can be obtained by applying this move to a user class.

3.2.2 Server optimization

This sub-problem is the optimum allocation of server applications to server machines. Server applications are organized in tiers, i.e., into sets of server applications that cooperate to serve the same request (see Table 2). Each server application or application tier has to be assigned to exactly one server (or server farm). The problem can be modelled as a *Set Covering Problem* (Papadimitriou and Steiglitz 1982; Ardagna et al. 2002).

3.2.3 Server localization

This sub-problem is the optimum allocation of servers to sites. Two cost items are affected by the allocation of servers: WAN costs and hardware support personnel costs. This cost-minimization sub-problem can be modelled as a network optimization problem which, in turn, can be represented as a min k-cut problem (Lengauer 1990; Ardagna et al. 2002). Since this problem is strongly NP-hard, a heuristic approach based on local search is adopted. The neighborhood of each feasible solution is defined by all solutions that can be obtained by moving a server to a different site. The search is guided by a tabu-search meta-heuristic in which only the short-term memory mechanism has been implemented.

3.2.4 Reuse of legacy systems

Each site has a (possibly empty) set of legacy machines and a set of servers defined by previous optimization steps. Each server could be replaced by one or more combinations of legacy machines. Moreover each legacy machine could be upgraded to provide higher performance. This problem can be modelled as a *Set Packing Problem* (Papadimitriou and Steiglitz 1982; Ardagna et al. 2002).

Different from legacy servers, legacy clients are supposed to be assigned to the user class that owns them. Therefore, decisions on the reuse of legacy clients are simply made by comparing the cost of upgrading legacy clients to meet current requirements with the acquisition costs of new machines.

3.2.5 Overall re-optimization

The decomposition of the overall optimization problem into four sub-problems does not guarantee that the final solution is a global optimum. Hence, an overall re-optimization process based on a tabu-search approach has been implemented to improve the (possibly) local optimum obtained by separately solving the four sub-problems. The move that is applied is defined as follows. A user class, say C_i, or a server application, say A_i, is disconnected from the server, say $server_A$, to which is currently connected. A new minimum-cost server (or server farm), say $server_B$, is selected to replace $server_A$. Costs are evaluated by assuming that $server_B$ is located in the same site, say S_A, of the replaced server. A new minimum-cost server, say $server_C$, is introduced to support C_i (or A_i), which is selected by comparing the cost of allocating the server in each site different from S_A. For each site, server management costs and network communication costs are evaluated. In this way, a destination site, say S_B, is identified for $server_C$. At last, the possibility of discarding $server_C$ is evaluated by connecting C_i (or A_i) to a different server in S_B. For each considered server a new upgraded configuration is evaluated on the basis of both acquisition and management costs. The neighborhood of a solution is defined by all solutions that can be obtained by applying this move to all user classes and to all server applications sharing a server. The search is guided by a tabu-search meta-heuristic in which only the short-term memory mechanism has been implemented.

4. Empirical Verifications

The methodology is verified to assess the magnitude of cost reductions and, thus, the potential benefits of a systematic methodological approach to cost issues of infrastructural design. Analyses were supported by a prototype tool that implements the methodology presented in Section 3 and integrates CPLEX, a commercial tool for integer linear programming, with meta-heuristic algorithms (see Figure 2).
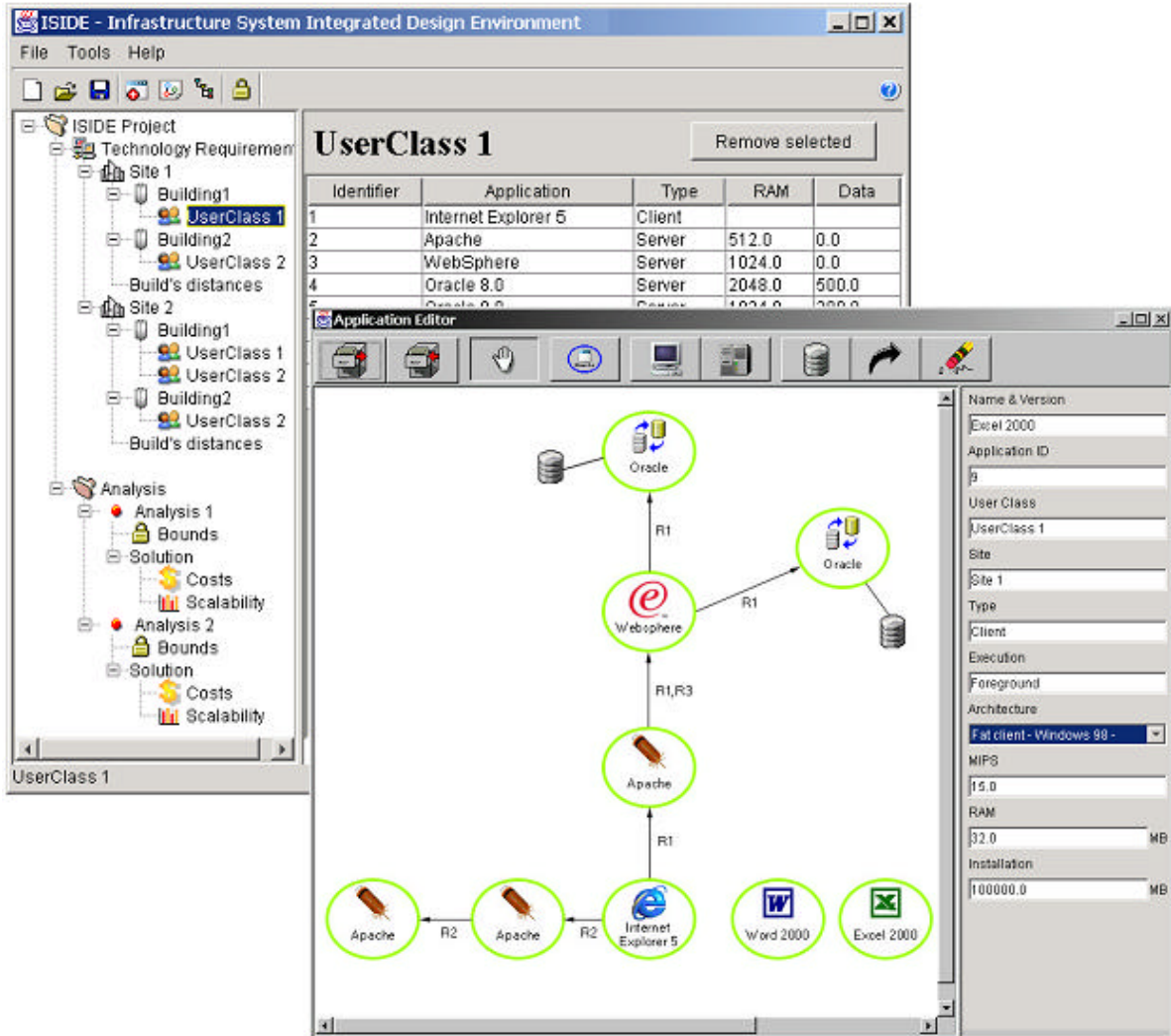
Figure 2 – Sample window of the tool implementing the methodology.

Empirical verifications have been based on the following sample of physical components and related cost data:

- 80 thin client configurations from 5 major vendors.

- 1100 PC configurations from 4 major vendors.

- 9000 server configurations from 4 major vendors.

- 1000 switch configurations and 10 hubs form 3 major vendors.

- 50 backbone link node router configurations form 3 major vendors.

Cost reductions are assessed for all design alternatives listed in Table 1. Individual design alternatives listed in Table 1 are analyzed first, by separately modifying corresponding technology requirements variables.   Variables are then simultaneously modified to test whether local results change as an overall design perspective is taken. Savings are evaluated as follows:

- Tests on client topology, number of tiers, allocation of applications and server location: average savings are calculated as the mean value of cost variations between the minimum-cost solution and all viable alternatives.

- Tests on total number of servers: average savings are evaluated as the mean value of the difference between the cost of the optimum solution and the cost of neighboring solutions with either one additional or missing server in the server farm.

- Tests on reuse of legacy: average savings are evaluated as the mean value of the difference between the cost of the optimum solution and the cost of a corresponding solution obtained by reusing all legacy machines possibly upgraded.

- Overall optimization: average savings are evaluated as the mean value of the difference between the cost of the optimum solution and the cost of a corresponding solution obtained by applying the following professional guidelines according to the professional literature:

  o Centralizing all application servers at one site, to minimize management costs.

  o Allocating applications with the maximum number of tiers, to minimize hardware acquisition costs.

  o Introducing server farms built with the smallest server in the database, to minimize hardware acquisition costs.

Significant cost reductions have been obtained in all cases, with average 30% savings. Findings are summarized in Table 5 and contrasted against the design rules of the professional literature. Results show that general design guidelines are difficult to infer from empirical results, thus challenging current professional rules.

Architectural alternative	Findings
Client typology	Results show that for a low number of users (1-6), PCs are selected as the cost-minimizing type of client. For a number of users higher than 6, the cost-minimizing solution switches to thin clients and does not further modify as the number of users grows. Savings from the adoption of HFCs range between 2 and 5%. This low value contradicts previous results in the professional literature, which present HFCs as a potential source of significant cost reductions (Molta 1999b; Microsoft 2000a).
Number of tiers	The number of tiers minimizing costs increases with requests' frequency, consistent with the professional literature (Dewire 1997). However, the number of tiers minimizing costs is found not to increase linearly with requests' frequency.
Total number of servers	Given the discrete distribution of physical servers along capacity requirements, the number of servers increases discontinuously with a varying capacity of servers in the server farm as the total computing requirements increase. This partly contradicts the professional literature which suggests always to introduce smaller machines to obtain cost reductions (Scheier 2001).
Allocation of applications (or server sharing)	Either a single-server or a multi-server option was preferred depending on total load. Based on professional guidelines, the multi-server solution would be expected to provide cost reductions above a load threshold. Results challenge the generalizability of this design rule.
Reuse of legacy	Reusing and possibly upgrading legacy machines minimizes costs in 65% of all tests, consistent with the professional literature (Bisbal et al. 1999).
Location of servers	The centralized solution is preferred in most, but not all cases (54% of all tests). In this respect, the professional literature provides design guidelines generally supporting the centralized solution (Robert Frances Group 1999, HP 2002).
Overall optimization	The centralized solution is the optimal solution in 30% of all tests. The maximum number of tiers is the optimal solution in 40% of all tests, while the number of tiers of the minimum-cost solution varies with no recurring pattern, contrary to professional guidelines (Dewire 1997; Scheier 2001).

Table 5 – Summary of findings.

5. Conclusions

Preliminary results from the empirical verification of the methodology indicate that cost reductions can be significant. Cost reductions are substantial when design alternatives are considered in isolation and do not decrease when design alternatives are combined within a more complex design problem of a practical infrastructure. This indicates that cost variations on individual design alternatives do not counterbalance each other and the *divide et impera* paradigm does not seem helpful from a cost perspective. Current work is completing the range of design alternatives by extending the methodology to include network design alternatives.

References

Ardagna D and C Francalanci (2002), 'A cost-oriented methodology for infrastructural design', Politecnico di Milano, Dipartimento di Elettronica e Informazione, Technical Report n. 2002.22, June 2002.

Ardagna D, Francalanci C, Trubian M (2002), 'A multi-model algorithm for cost-oriented infrastructural design', Politecnico di Milano, Dipartimento di Elettronica e Informazione, Technical Report n. 2002.47 November 2002.

Aue, A and M Breu (1994), 'Distributed Information Systems: An Advanced Methodology', IEEE Trans. on Software Engineering, 20(8), 594-605.

Bisbal, J, Lawless, D, Bing Wu, Grimson, J (1999), 'Legacy information systems: issues and directions', IEEE Software, 16(5) 103 –111.

Dewire D T (1997), 'Second-generation Client/Server Computing', McGraw Hill.

Ein-Dor, P (1985), 'Grosch's Law Revisited: CPU Power and the Cost of Computing', Management Communication of the ACM, 28(2), 142-150.

Faye Borthick, A and HP Roth (1994), 'Understanding Client/Server Computing', Management Accounting, Aug., 36-41.

Gartner Group (2002). TCO Manager. <http://gartner11.gartnerweb.com/bp/static/tcomanhome.html>.

Gavish, B and H Pirkul (1986), 'Computer and Database Location in Distributed Computer Systems', IEEE Trans. on Computers, 35(7), 583-590.

Grosch, HA (1959), 'High Speed Arithmetic: The Digital Computer as a Research Tool', Journal of Operational Research 43(4).

Harchol-Balter, M and AB Downey (1997), 'Exploiting Process Lifetime Distributions for Dynamic Load Balancing', ACM Transactions on Computer Systems. 15(3), 253-285.

HP (2002). The scope of HP systems consolidation. <http://www.hp.com/products1/unixservers/solutions/sc/scope.html>.

Jain, HK (1987), 'A comprehensive model for the design of distributed computer systems', IEEE transactions on software engineering, 13(10), 1092-1104.

Lazowska, ED, Zahorjan, J, Graham, GS, Kenneth, CS (1984), 'Quantitative System Performance Computer system analysis using queueing network models', Prentice-Hall.

Lengauer, T (1990), 'Combinatorial Algorithms for Integrated Circuit Layout', John Wiley & Sons, Chichester.

Lui, JCS, and MF Chan (2002), 'An efficient partitioning algorithm for distributed virtual environment systems', IEEE Transaction on Parallel and Distributed Systems, 13(3), 193-211.

Main Control (2002). Technology Asset Management. <http://www.maincontrol.com>.

Menascé, DA and VAF Almeida (2000), 'Scaling for E-business, Technologies, models, performance and capacity planning', Prentice-Hall.

Microsoft (2000a). Windows 2000 Terminal Services Capacity and Scaling. <www.microsoft.com/windows2000/library/technologies/terminal/tscaling.asp>.

Microsoft (2000b). Microsoft Application Center. <http://www.microsoft.com/applicationcenter/evaluation/overview/default.asp>.

Moad, J (1994, 'Client/Server Costs: Don't Get Taker for a Ride', Datamation, 15(2), 34-41.

Molta, D (1999a), 'For Client/Server, think thin', Network Computing, www.networkcomputing.com/1013/1013f1.html.

Molta, D (1999b), 'Thin Client computers come of age', Network Computing,

<www.networkcomputing.com/1009/1009buyers1.html>.

Papadimitriou, C and K Steiglitz (1982), 'Combinatorial Optimization', Prentice Hall, Inc., Englewood Cliffs, New Jersey.

Scheier, RL (2001), 'Scaling up for e-commerce', Computerworld, <http://www.computerworld.com/softwaretopics/software/appdev/story/0,10801,59095,00.html>.

Stevens, L (1997), 'Strategies for Distributed Systems', <http://www.uniforum.org/news/html/publications/ufm/jan97/thin.html>.

The Robert Frances Group (1999). Server consolidation strategies. <http://www-1.ibm.com/servers/solutions/serverconsolidation/pdf/robertfrancesgroup.pdf>.

The Tolly Group (1999). Total Cost of Application Ownership. <www.tolly.com>.

Willcocks, L (1992), 'Evaluating Information Technology Investments: Research, Findings, and Reappraisal', Journal of Information Systems, 2, 243-268.