Characterizing a Synthetic Workload for Performance Evaluation during the Migration of a Legacy System*

Paulo Pinheiro da Silva † Alberto H. F. Laender ‡ Rodolfo S. F. Resende ‡
Paulo B. Golgher ‡

† Department of Computer Science
University of Manchester
Oxford Road, Manchester M13 9PL, UK.
pinheirp@cs.man.ac.uk

‡ Department of Computer Science
Federal University of Minas Gerais
31270-010 - Belo Horizonte MG - Brazil
{laender,rodolfo,golgher}@dcc.ufmg.br

Abstract

This paper describes the characterisation of a synthetic workload for performance evaluation of a new system before replacing a legacy system. The workload is used by CAPPLES, a capacity planning and performance analysis method for the migration of legacy systems. Typical workload characterisation problems are anticipated and discussed. Further, guidelines to characterise a CAPPLES workload for different migration scenarios are provided.

1 Introduction

In most organisations, maintenance is a main concern after a system is deployed and enter in the production phase. Through maintenance, an organisation tries to keep its mission-critical systems up-to-date with new requirements. However, a situation can be reached where maintenance is not the best choice and a *legacy system*, i.e. an old system, is replaced by a new one.

The migration to a new system is an important and complex process. We are interested in the phase where the legacy system is still operational and the *target system*, i.e. the new system, is not in production but it is already developed.

Although there are many strategies [4, 5, 16] and tools [8, 14] to assist the migration of legacy systems, none of them provides mechanisms for evaluating the performance of the resulting target system. To the best of our knowledge, we believe that CAPPLES[1] [6, 7] is the only method suitable for the performance assessment of target systems during the migration of legacy system.

The CAPPLES method basically consists of simulating a synthetic workload composed of parameters collected from the operational legacy system and the developed but not operational target system. The use of synthetic workloads for the evaluation of new systems is not a new idea. For instance, much work has been done in order to identify how to characterise and generate workloads aimed at the evaluation of Web based systems [2, 3] and parallel and distributed systems [1, 15].

In [7], we describe the basic steps involving CAPPLES and give a brief overview of how to characterise a synthetic workload for performance evaluation during the migration of a legacy system. In this paper, we describe in detail the characterisation of a synthetic workload for a real case to which CAPPLES was applied. Typical characterisation problems such as the identification of the workload composition and intensity are anticipated and discussed through the use of a case study.

The workload case study refers to a successful performance evaluation study of a real legacy system migration. Both systems (the target and legacy ones) had more than 1 million lines of code and correspond to an application that is responsible for the control of the administrative and operational activities of a public water and sewage company in Brazil with more than 5 million customers, distributed over more than 500 cities. However, the strategy used in this case study and the proposed guidelines are generic enough to help the performance evaluation of other migration efforts.

The remainder of the paper is organised as follows. Section 2 describes the workload characterisation context. Section 3 describes how to identify and select a representative set of transactions in the legacy system to compose the workload. Section 4 describes how to identify services

*This work was supported by COPASA-MG under cooperative agreement 3025.

[1]CAPPLES stands for Capacity Planning and Performance Analysis Method for Legacy Systems

from the selected transactions. The service characterisation is discussed in Section 5. Section 6 presents the resulting synthetic workload. Section 7 presents the simulation results obtained using the characterised workload. Finally, Section 8 concludes the paper.

2 Workload Characterisation Context

This section starts with some definitions followed by an overview of CAPPLES.

An application can be modelled as a hierarchy of *tasks*, each one with a specific goal. Tasks can be decomposed into subtasks, that can be decomposed in subsubtasks and so on. Thus, tasks can be specified at different levels of abstraction. *Composite tasks* are decomposed into subtasks. *Primitive tasks* do not have subtasks. Using the task concept, it is possible to define the terms *service*, *on-line transaction* and *on-line service*.

A *service* is an high-level task composed of a set of conventional actions performed by their subtasks over the system. As a task, a service also has a specific goal. An *on-line transaction* is a primitive task that represents an interaction of the user with the application. This user interaction is performed through the use of devices such as displays, keyboards and mice, and results in an action execution. An *on-line service* is a service whose primitive tasks are on-line transactions. An on-line service may have one or many on-line transactions.

Considering these definitions, we can briefly explain CAPPLES. In the method, the synthetic workload is mainly composed of on-line transactions of the target system. CAPPLES was devised for evaluating the performance of target systems running services of any category, i.e. batch jobs, on-line transactions, interactive requests, etc. However, CAPPLES can show its usefulness more effectively in systems where on-line services are suspect to have performance problems [7].

The target system on-line transactions that compose the synthetic workload correspond to the most frequently executed transactions of the legacy system. However, a given transaction of the legacy system may not have a corresponding transaction in the target system or, when it exists, it might be difficult to identify. Therefore, the ideal transaction mapping shown in Figure 1 is usually not feasible. In CAPPLES, the corresponding transactions of the legacy and target systems are identified through an indirect mapping of the transactions using the concept of service, as defined above. This concept is discussed throughout this paper. Unfortunately, our case study also uses the term *service* with a different meaning (i.e., Service Order of a water company, such as a water leaking service). We hope the context will make it clear what meaning we intend.

The indirect transaction mapping, shown in Figure 1, is a feasible approach to identify corresponding transactions since services are usually easy to identify and match in both systems. For example, it is easy to find out how to perform a `Customer Information Updating` in both systems. However, it might be difficult to identify how `Customer Information Updating` transactions can be directly put in correspondence.

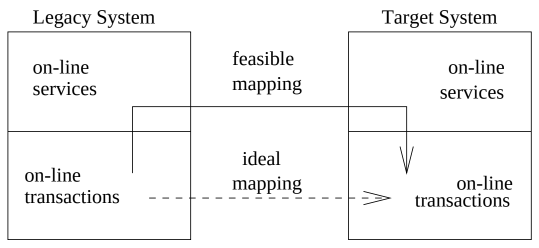

Figure 1. Indirect mapping of transactions.

In order to generate a generic workload, it is necessary to identify the workload services. Moreover, for each service it is necessary to identify its intensity and resource demands [12]. In CAPPLES, however, in order to generate a workload it is also necessary to: (1) identify the most frequently executed transactions of the legacy system; (2) identify in the target system the corresponding transactions of the legacy system; and (3) characterise the intensity and resource demands of the on-line transactions. These workload characterisation activities are described in detail in the following sections.

3 Identifying and Selecting Legacy System Transactions

An *on-line time window* is a period of time that is dominated mainly by the execution of on-line transactions. It is also characterised by the fact that on-line transactions have higher priority than other activities. An on-line time window usually corresponds to a period of time over the organisation working hours. For instance, a supermarket system on-line time window usually include weekend days, while a financial organisation on-line time window does not include weekend days.

The most frequently executed legacy system on-line transactions are identified by measuring the system usage during its on-line time window. While legacy systems can be monitored for a long period of time, its workload profile, in terms of the most frequently executed on-line transactions, tends to be the same week by week. Exceptional situations such as a stock-market crash in a financial company can affect this assumption, for example. However, these situations can be easily identified, and a general guideline can be provided concerning legacy system measurements.

Guideline 1 *The legacy system parameters that are required for the workload characterisation are the category, the executed program*[2] *and the time of execution of each on-line transactions. Transaction execution frequencies can be obtained from the consolidation of the collected parameters.*

Concerning Guideline 1, many other measures (e.g. CPU cycles and I/O accesses) could usually be collected from transaction monitor tools. However, as the target system is new and probably built over new technologies, most of the measures might not be representative in the new system.

In our case study, the legacy system on-line services are mainly composed of CICS activated COBOL programs. Therefore, a CICS monitor was used to measure the frequency of execution of the on-line transactions, as suggested by Guideline 1. Table 1 shows a partial view of the measured transactions execution frequencies consolidated by their program names. The complete table of the measured transactions has 2,982 entries.

We define the *measuring time window* as a period within the on-line time window when the legacy system is monitored in order to determine the most executed on-line transactions.

Transaction Category Name	Program Name	Frequency
PRNT	PGP0473	208464
HPSX	VHP07421	117403
HPSX	VHP07428	64528
HPSX	VHP07426	58854
HPPX	VHP07470	50283
SATX	ST123100	49616
HPAT	PHP07401	45592
...	...	...

Table 1. Partial view of the consolidated transaction frequencies during the measuring time window.

After the measurements, it must be determined which transactions will be used to compose the synthetic workload. In our case study, we selected the 24 most frequently executed transactions. These selected transactions were the *relevant on-line transactions* of our case study legacy system. The selection cutting point was based on the analysis of the execution frequency of each transaction. The inclusion of more transactions would contribute little to the representativeness of the selected set. The representativeness of a set of transactions corresponds to the ratio between the execution frequency of the transactions in the set and the execution frequency of all transactions. For instance, the inclusion of the next most executed transaction (i.e., the 25th) would increase the set representativeness by only 0.6%.

[2]A program can have many denominations in terms of measured parameters: routine, screen, class method, etc.

As the target system will be a mission-critical system when operational, it cannot stop any time during the on-line time window. Therefore, the workload should model only the worst scenario in order to identify if the target system will cope with its real workload. In this case, the *simulation time window*, defined as the legacy system usage peak hour (or fraction of the peak hour) during the measured time window, is the worst scenario. The peak hour can be identified consolidating the measured transactions by their time of execution.

In our case study, the transactions measured in the simulation time window were further classified by geographical regions. This was necessary due to the distributed nature of the case study target system. While this information is not important to understand the synthetic workload generation process, it is needed to keep the consistency of the data presented in the case study. Indeed, in what follows the tables only reflect values of one region. Table 2 shows a partial view of the relevant transactions ordered by their frequencies in one of those regions.

The `PRNT` transaction in Table 1 corresponds to printing services, which is a batch job service in the target system. Most of the analysis required to generate the final workload does not apply to batch services. Therefore this batch job will be included in the final workload later, as described in Section 6.

Transaction Name	Program Name	Frequency
HPSX	VHP07426	196
HPSX	VHP07421	184
HP1C	PHP07443	167
HPSX	VHP07428	130
SATX	ST123100	123
HPSX	VHP07424	109
...	...	...

Table 2. Relevant on-line transactions measured during the simulation time window.

4 Identifying Services

Having identified the relevant transactions in the legacy system, it is necessary to identify their related services. A thorough analysis of the most executed transactions and its corresponding programs, along with interviews with experienced legacy system users, should suffice. Sometimes, the source code of the legacy system will have to be analysed. The purpose of this analysis is to map the set of program and transaction names obtained during the measuring time window into conventional actions performed in the legacy system. Thus, for example, the program `VHP07421` is known to be the initial screen of the `Service Order Entry` service, the program `VHP07424` is a help screen to locate the correct customer block, and the program `VHP07428` effectively inserts `Service Orders` in the database. Thus, the set {`VHP07421`,`VHP07424`,`VHP07428`} can be considered as the complete `Service Order Entry` service, as shown in Table 3.

Service	Transaction	Description	Freq.
Service Order Entry	VHP07421	Customer Identification	184
	VHP07424	Browse Customer Block (help)	109
	VHP07428	Generate the Service Order	130
Customer Information Report	VHP07470	Customer Identification	83
	VHP07472	Generate Report	73
Customer Information Updating	PHP07411	Customer Identification	46
	PHP07403	Information Updating	23
Service Order Finishing	ST123100	Mark a Service Order as completed	123
...	...	...	...

Table 3. Partial view of identified on-line services.

Certainly, the frequency of execution of these programs will not be equal, and the analysis of the differences between these measures provides an insight of how the legacy system was used. For instance, in the former example, the program `VHP07421` was executed 184 times while the program `VHP07428` was executed 130 times. Probably, the user cancelled the generation of the service order 54 times. Further, the differences between the frequency of execution of the initial screen and the program `VHP07424`, indicates how many times the service was executed using the help subsystem. This sort of analysis is described in Section 5.

Programs may be related to different transaction categories. Therefore, programs can be performed in different contexts. Thus, the service identification task must take both transaction categories and programs in consideration. Table 4 shows a partial list of the resulting services in our case study. The 24 selected transactions(partially listed in Table 2) resulted in 12 services.

Service Name	Description	Frequency
so-entry	Service Order Entering	184
so-finish	Mark a Service Order as completed	123
so-upd	Service Order Updating	84
cust-rpt	Customer Information Reporting	83
so-query	Service Order Querying	68
so-rec	Service Order Receiving	67
cust-upd	Customer Information Updating	46
db-query	Customer Debits Querying	29
cust-inf	Customer Information Querying	28
cso-query	Commercial Service Order Querying	28
bd-rpt	Customer Debits Reporting	19
ed-entry	Extra Debits Entering	10

Table 4. Identified on-line services.

These services will be used as the link between the knowledge developed around the legacy system and further data drawn from the target system that will be used to characterise the transactions in the synthetic workload. The frequency of execution of these services will be used to determine the frequency of execution of the equivalent services in the target system, and therefore to characterise the volume of services in the target system. Table 5 summarises the parameters required to characterise the transactional part of the synthetic workload.

Parameter	Source System	Mandatory
Generic transactional workload parameters		
Transaction composition	legacy system	yes
Transaction ordering	legacy system	optional
Component transaction parameters		
intensity	legacy system	yes
memory demand	target system	optional
service CPU demand and DBMS CPU demand	target system	yes
service I/O demand and DBMS I/O demand	target system	optional
local area network demand	target system	optional
wide area network demand	target system	optional

Table 5. Parameters of a transactional workload.

5 Service Characterisation: Service Use Cases

Services are abstractions of their subtasks where actions and user interactions effectively happen. For this reason, there are usually many ways to carry out a service, each one demanding a different amount of computer resources. This section describes how to identify service use cases and service use case participation in services' intensities.

5.1 Use Case Identification

Hierarchy task analysis [10, 11] (HTA) can be used to describe how services are composed in terms of tasks. Figure 2 presents a partial representation of the task hierarchy of the `customer information updating` service (`cust-upd` in Table 4). As a composite task, a service can be decomposed into primitive tasks and other composite tasks. If primitive tasks are always executed, then there is only one way to execute the service. On the other hand, if there are primitive tasks that may not be executed, then there is more than one way to execute the service. Considering this task analysis, each available combination of service primitive tasks identifies a service use case. In fact, a service has at least one use case.

Using the service use case concept, we can identify the use cases for the services in Table 4. Service task hierarchies can be complex and hard to identify. It is important to remember that the identification of service task hierarchies is a reverse engineering activity, even for the target

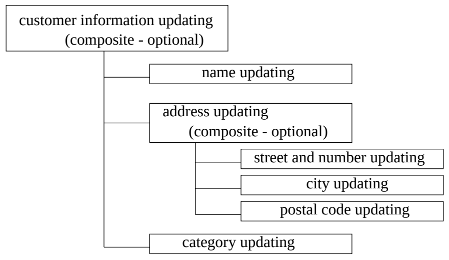
customer information updating (composite - optional)
name updating
address updating (composite - optional)
street and number updating
city updating
postal code updating
category updating

Figure 2. The task hierarchy of the customer information updating service.

Service	Use Cases Names	Description
Service order entry	so-entry-nv so-entry-ex so-entry-add	Service Order Entry - naive user Service Order Entry - expert user Entry using Customer Address for identification
Customer information report	cust-rpt-nv cust-rpt-ex cust-rpt-det	Customer Report - naive user Customer Report - expert user Detailed Customer Report
Customer information update	cust-upd-name cust-upd-add-nv cust-upd-add-ex cust-upd-catg-nv cust-upd-catg-ex	Name updating Address updating - naive user Address updating - expert user Category updating - naive user Category updating - expert user
...	...	...

Table 6. Identified service use cases.

system. Further, it is possible that some legacy code inspection should be required.

Moreover, our workload characterisation is impacted if users interact with the application using help facilities such as combo boxes, search routines, and choose routines. In the case of the use of combo boxes, for example, the user interaction can require additional processing to obtain their items. Therefore, there may be many possible ways to perform a service consuming different resources. The less time consuming way is classified as an *expert* use case, while the more time consuming way is classified as a *naive* use case. This additional classification does not invalidate the fact that each available combination of service primitive tasks identifies a service use case. Indeed, further decompositions of the original primitive tasks lead to new composite and primitive tasks where the help facilities would be considered.

There are two guidelines concerning the identification of service use cases.

Guideline 2 *Only the children tasks of the tasks that represent services should be considered during the identification of an use case.*

In fact, as a synthetic workload, it should be similar to a real workload, but it may not be a real workload. Further, CAPPLES was designed to be applied in a short period of time. It means that the workload characterisation must be executed in the shortest possible time.

Guideline 3 *Two use cases should be considered for user interfaces composed of many interactors providing help facilities such as combo boxes: one describing the interaction of a typical expert user that does not use any help facilities, and one describing the interaction of a typical naive user that uses the available help facilities.*

Table 6 shows the use of these guidelines on the 12 services presented in Table 4. In our case study, the 12 services expanded to 38 use cases.

5.2 Use Case Intensity Participation

Once identified the use cases, we need to stipulate the participation of each use case in the total number of executions of the service. For example, we know that the *customer information updating* was executed 46 times during the simulation time, but we do not know how many of these 46 times were executed to update the customers' address, name or category.

Fortunately, the `customer information updating` service is one of the most frequent services performed in this system. Therefore, it is expected that the organisation keeps record of which percentage of these `customer information updatings` modifies customers addresses, and which percentage modifies customer categories. Considering, for example, that the organisation knows that 82% of customer information updatings are address and name updatings, and 16% are customer category updatings, we can identify the participation of most use cases. A generic approach to identify use case participation can be introduced using the *business unit* concept. A business unit corresponds to a variable that can be used as an indication of the volume of activity of a business or administrative function [12].

The assumption adopted here is that: (1) high-frequent services are related to well-known business units; and (2) organisations keep record of their business units. The `customer information updating` service is an example of how business units can be used to identify use case participation. Customers are natural business units for most organisations. Therefore, the organisation knows details about their customers.

However, it is not expected that an organisation keeps record of how system users interact with the user interface. For example, the organisation does not know how many times system users directly provide a code or use a combo box to fill out customers' categories.

There are four guidelines concerning the identification of the participation of each use case in a service.

Guideline 4 *Use business unit statistics when available since they are long-term statistics.*

Guideline 5 *Use legacy system measured transaction frequencies when business statistics are not available.*

Guideline 6 *If a service has more than one use case and there are no business unit statistics available, then the identified use cases can be considered to have the same participation.*

Guideline 7 *In systems where most users are frequent ones, use cases related to expert users can be estimated to represent 80% of the original use cases that were classified into* expert *and* naive *use cases.*

There are two aspects of the service characterisation that should be considered before the integration of the workload parameters: interaction cancelling and service submission ordering.

5.3 Interaction Cancelling

User interfaces usually have interactors such as buttons or commands that allow users to cancel (or quit) an on-line service [13]. The use of cancelling interactors generates new service use cases. Non-cancelled use cases demand a different amount of computer resources as their respective cancelled use cases. For instance, the cost to commit the database transactions associated with a non-cancelled use case should be different than the cost to rollback the database transactions in a respective cancelled use case.

Cancelled use cases implicitly appear in the measured programs in the legacy system. In Table 2, the program `VHP07421` is the initial user interface where users identify the customer which is requesting a service order. Program `VHP07428` actually generates the service order. The frequency of execution of the program `VHP07428` is 130, that is roughly 70% of the frequency of the program `VHP07421`. Therefore, there is 30% of cancelled use cases for the `Service Order Entry` service. This high percentage might imply the necessity of adding another use case, `so-entry-canc`, that represents the cancelled interaction.

This sort of analysis can be further carried out in the lower levels of the task hierarchy. Thus, composite subtasks may have implicit cancelled use cases that must be modelled in the final workload. This leads to the following guidelines.

Guideline 8 *Transaction frequencies of the measured time window are better representations of the average behavior of the real system than the transaction frequencies of the simulation time window.*

Guideline 9 *Cancelled use cases should be obtained from the transaction frequencies.*

Target systems usually have a completely redesigned interface and perform the organisation activities in a different way. This can result in significant differences between the legacy and target systems interaction cancelling profile. Thus, only activities that are prone to have cancelled interactions should have their cancelled use cases modelled. This leads to the third guideline concerning interaction cancelling issues.

Guideline 10 *Cancelled use cases with low frequencies (less than 10%) should be ignored.*

5.4 Service Submission Ordering

In the case study, only non-executed operational services (OS) can be marked as executed. Therefore, the `so-entry` service, that registers new OSs, and the `so-finish` service, that marks the OSs as executed, are temporal dependent on each other. Formally, we say that a service Y has temporal dependency with service X if service Y can only be executed after the completion of service X.

However, the study is not considering all possible temporal dependencies that might exist between services, as stated below.

Guideline 11 *Only temporal dependencies that occur during the simulational time window should affect the workload characterisation.*

Therefore, the `so-entry` and `so-finish` services are not considered as temporal dependent. Indeed, the OSs marked as executed by the `so-finish` service are not the same OSs that were registered into the application database by the `so-entry` services during the simulation time window. One hour is not long enough for the simulation time window to allow a single operational service to be recorded, executed, and marked as executed.

A solution can be devised for the problem of possible temporal dependencies in other migration scenarios. Another guideline may be considered to explain how temporal dependencies affect workload characterisations.

Guideline 12 *Temporal dependencies are used instead of service submission distribution in the composition of the synthetic workload.*

The identification of all temporal dependencies between selected services, associated with the use of the Guidelines 11 and 12, can be a naive approach to address the temporal dependency problem. The use of the Guideline 12 is explained in the next section, where the last activities are described in order to characterise the synthetic workload.

6 The Synthetic Workload: Putting All Together

The use case resource demands presented in Table 5 should be measured in the target system to complete the synthetic workload characterisation. These remaining parameters can be measured using traditional techniques [9] and accounting tools. Table 7 shows the characterisation of the `Customer Information Report` use cases. In this case, legacy data was not available to infer use case participation. Thus, the participation was initially equally distributed among the three use cases, as suggested in Guideline 7, which resulted in one third of the participation for each use case. Then, the two thirds corresponding to the naive and expert use cases were divided following Guideline 8, where the expert use case received 80% of the corresponding fraction of the participation, which yielded 53.3% for the expert use case (80% from 66.6%) and 13.3% for the naive use case (20% from 66.6%).

Parameter	cust-rpt-nv	cust-rpt-ex	cust-rpt-det
Participation (%)	13.3	53.3	33.3
CPU demand (ms)	1670	770	2110
I/O demand (operations)	180	101	279
LAN demand (packages)	151	53	131
WAN demand (frames)	2718	877	2030

Table 7. Customer information updating service composition.

Considering the participation of each use case, the average of these parameters should compose the parameters of the transactional workload. When the participation is equally distributed among the use cases, Jain [9] suggests that the average should be used only if the values of the measured parameters are homogeneous. If not, the median of the values should be used.

In the final workload, services are submitted according to a Poisson distribution. The intensity parameter is used as the distribution argument. If the service depends on other services it should be submitted according to the submission distribution of the services which have temporal dependencies, and only after they have been completed. Table 8 shows a partial view of the transactional workload for our case study.

The final step in the generation of the synthetic workload is the addition of non-on-line-services identified during the simulation time window. As CAPPLES is aimed at the evaluation of transactions of the on-line time window, as described in [7], only few batch jobs are expected. The process of identifying the equivalent batch jobs in the legacy and target systems is similar to the process applied to on-line services, but simpler because use case analysis is not required. In our case study, two batch jobs were considered and modelled in the synthetic workload: (1) printing services and (2) *ftp* file transfers. Table 9 shows the measured parameters for these non-on-line-services.

Parameter	so-entry	cust-rpt	cust-upd	so-finish
Intensity(%)/Subm. ordering	24	11	6	16
memory demand (MBytes)	NR	NR	NR	NR
CPU demand (ms)	2070	1335	880	1950
I/O demand (operations)	262	171	61	153
LAN demand (packages)	222	92	108	197
WAN demand (frames)	4290	1504	1692	3231

Table 8. Partial view of the transactional workload.

Parameter	Jobs batch	ftp
memory demand (MBytes)	NR	NR
CPU demand (ms)	13.75	256.0
I/O demand (operations)	14	1320
Printer demand (ms)	4653.3	NR
Number of ethernet packages (qtd.)	NR	183
Size of ethernet packages (byte)	NR	1500
Number of X.25 frames (qtd.)	NR	2286
Size of X.25 frames (byte)	NR	120

Table 9. The non-transactional workload.

7 Experimental Results

In order to assess the guidelines and techniques presented throughout this paper, and to show the potential benefits of having a well-characterised workload, we present some experimental results taken from our study case. These results were obtained using the CAPPLES method, which also encompasses a simulation model of the target system and the use of forecasting techniques.

The comparison of the mean response time of the on-line transactions in the target and legacy systems is one of the most important results in order to validate the target system performance. Figure 3 shows the predicted results.

These results, compared to the legacy system's 900 ms mean response time, were good news to the organisation in terms of performance. However, in other migration efforts, this analysis could show that the target system was not well-dimensioned before it was operational, allowing the organisation to take proper actions to guarantee the current standards of its services.

8 Conclusions

This paper discussed the characterisation of a synthetic workload that was used to evaluate the migration of a real legacy system using CAPPLES. The synthetic workload refers to an expected workload generated by a distributed

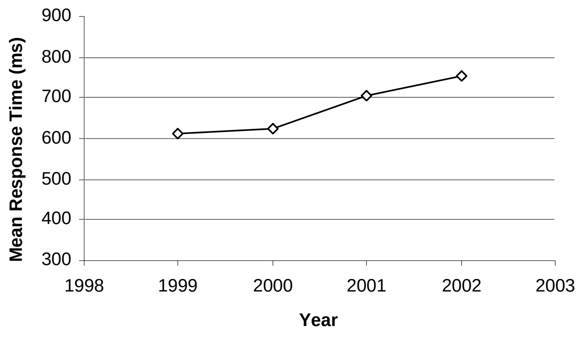

Figure 3. Predicted mean response time of the target system on-line transactions

application. The application is a large system that uses a database and network facilities. The techniques used to characterise this workload can be used to characterise other synthetic workloads for CAPPLES. Several of the assumptions and guidelines adopted in the presented study can also be used for evaluating the migration of other legacy systems in different environments.

The characterisation strategy used in this study can be summarised as follows. The most frequent on-line transactions of the operational legacy system were identified. Services were identified from these transactions and mapped into the target system. Target system on-line transactions were identified using the service use case concept. The participation of each use case in the number of on-line services to be submitted during the simulation time was identified. On-line transaction parameters were measured for each service use case. The transactional workload was characterised from the measured parameters collected from the target system and the intensity or submission ordering of each service use case. The workload generated by non-transactional services was obtained using traditional techniques. The final synthetic workload was composed of the transactional workload and the non-transactional workload.

Acknowledgements. The authors would like to thank Berthier Ribeiro-Neto and Virgílio Almeida for many valuable comments during this work. The first author is sponsored by Conselho Nacional de Desenvolvimento Científico e Tecnológico - CNPq (Brazil) – Grant 200153/98-6.

References

[1] W. Alexander, T. Keller, and E. Boughter. A workload characterization pipeline for models of parallel systems. *ACM SIGMETRICS Performance Evaluation Review*, 15(1):186–194, May 1987.

[2] M. Arlitt and C. Williamson. Web server workload characterization: The search for invariants. In *Proceedings of the ACM SIGMETRICS'96 Conference*, Philadelphia, PA, April 1996.

[3] P. Barford and M. Crovella. Generating Representative Web Workloads for Network and Server Performance Evaluation. In *Proceedings of the ACM SIGMETRICS'98 Conference*, pages 151–160, Madison, WI, July 1998.

[4] K. Bennett. Legacy systems: Coping with success. *IEEE Software*, 37(5):26–41, 1995.

[5] M. Brodie and M. Stonebraker. *Migrating Legacy Systems: Gateways, Interfaces, and the Incremental Approach*. Morgan Kaufmann, San Francisco, CA, 1995.

[6] P. Pinheiro da Silva. A Study on the Migration of Centralised Legacy Systems to a Distributed Enviroment. Master's thesis, Universidade Federal de Minas Gerais, Belo Horizonte, Brazil, June 1998. (In Portuguese).

[7] P. Pinheiro da Silva, A. Laender, R. Resende, and P. Golgher. CAPPLES - A Capacity Planning and Performance Analysis Method for the Migration of Legacy Systems. In P. Chen et al., editor, *Advances in Conceptual Modeling*, volume 1727 of *LNCS*, pages 198–212, Paris, France, November 1999. Springer.

[8] L. Markosian et al. Using an enabling technology to reengineer legacy systems. *Communications of the ACM*, 37(5):58–70, 1994.

[9] R. Jain. *The Art of Computer System Performance: Analysis, Techniques for Experimental Design, Measurement, Simulation and Modeling*. Wiley, New York, NY, 1991.

[10] P. Johnson. *Human Computer Interaction: Psychology, Task Analysis and Software Engineering*. McGraw-Hill, Maindenhead, UK, 1992.

[11] B. Kirwan and L. Ainsworth. *A Guide to Task Analysis*. Taylor & Francis, London, UK, 1992.

[12] D. Menascé, V. Almeida, and L. Dowdy. *Capacity Planning and Performance Modeling: From Mainframes to Cliente-Servers Systems*. Prentice-Hall, Englewood Cliffs, NJ, 1994.

[13] J. Nielsen. *Usability Engineering*. AP Professional, Boston, MA, 1994.

[14] J. Ning, A. Engberts, and W. Kozaczynski. Automated support for legacy code understanding. *Communications of ACM*, 37(5):50–57, 1994.

[15] R. Bunt R. Bodnarchuk. A synthetic workload model for a distributed system file server. *ACM SIGMETRICS Performance Evaluation Review*, 19(1):50–59, May 1991.

[16] I. Sommerville, J. Ransom, and I. Warren. A method for assessing legacy systems for evolution. Technical Report TR/07/97, Computing Department, Lancaster University, Lancaster, UK, 1997.