A Simulation Model for the Performance Evaluation for Migrating a Legacy System[*]

Paulo Pinheiro da Silva [†] Alberto H. F. Laender [‡]
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
31270-901 - Belo Horizonte MG - Brazil
{laender,golgher}@dcc.ufmg.br

Abstract

It would be strategically advantageous to organisations if they could predict the behaviour of the new systems meant to replace existing legacy ones. In fact, these organisations could tackle possible performance problems of the new systems before they become operational. This paper describes how CAPPLES, a capacity planning and performance analysis method for the migration of legacy systems can be used to predict such behaviour. The paper describes the process of constructing, validating and deploying a simulation model using CAPPLES. Peculiarities of the legacy system migration that affects the simulation process are presented through the use of a case study. Finally, guidelines to perform similar simulation studies are also provided.

1 Introduction

During the life cycle of a software system it is customary to carry out capacity planning and performance analysis, not only to evaluate the system's environment but also to plan its operation [5, 15]. This type of study would also be very useful when the migration of a legacy system is planned. However, traditional capacity planning and performance analysis methods [4, 7, 9] cannot cope with certain problems concerning the migration of a legacy system. The first problem is that the method must provide guidance on the performance of the new system on the basis of the current one. The second problem is that the legacy and target systems are usually very large, normally more than a million lines of code, which makes the performance analysis very complex.

CAPPLES, a CApacity Planning and Performance analysis method for the migration of LEgacy Systems [1, 2, 3] has been developed to address the particularities of performance evaluation during migration. CAPPLES is based on simulation models. Indeed, analytical models usually require simplifications that may not be adopted for mission-critical systems. Moreover, experiments may not be possible in an actual computing environment since resources (such as trained people and remote links) may be unavailable. Organisations would need to halt their essential activities to perform such evaluation.

This paper describes a complete simulation process used to evaluate the performance of a non-operational target system. Furthermore, it is explained how a synthetic workload generated from our case study's legacy and target systems (as described in [3]) is used in simulation activities. The case study details presented in this paper are complementary to those presented in [3]. Thus, performance analysts have a comprehensive description of how CAPPLES is used. This case study is based on the actual migration of a mission-critical system at COPASA-MG, the Water and Sewage Company of the State of Minas Gerais, Brazil.

This paper is organised as follows. A description of the simulation process is presented in Section 2. The case study used to illustrate the simulation process is introduced in Section 3. A description of how the simulation model was built is presented in Section 4. A description of how experiment results were obtained in order to validate the simulation model is presented in Section 5. The validation of the simulation model is described in Section 6. The use of the validated model to predict the behaviour of the target system is presented in Section 7. Conclusions about the presented study are presented in Section 8.

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

2 The Simulation Process

The submission of a synthetic workload to a simulation model provides a mean to predict the behaviour of a target system. Therefore, we need a way to specify and to execute this simulation model of the target system's computing environment. Simulators, such as SMPL and SES/Workbench [10], provide constructors to specify models, as they also implement techniques to execute the models.

The simulation process for performance evaluation of a target system is shown in the activity diagram in Figure 1. The challenging task in the simulation process is the production of a valid model. The model is mainly built in the `Modelling` activity where it is specified. In the `Simulation` activity the model is executed, producing simulation results. The `Validation` activity is based on a comparison of the simulation results with the results produced in the `Experimentation` activity. If the simulation and experimental results are similar enough, the model can be classified as valid. Otherwise, the `Modelling refinement` activity may be required to produce simulation results that correspond more closely to the experimental results. Once the model is validated, the `Prediction` activity is executed producing the simulation results that describe the predicted behaviour of the target system.

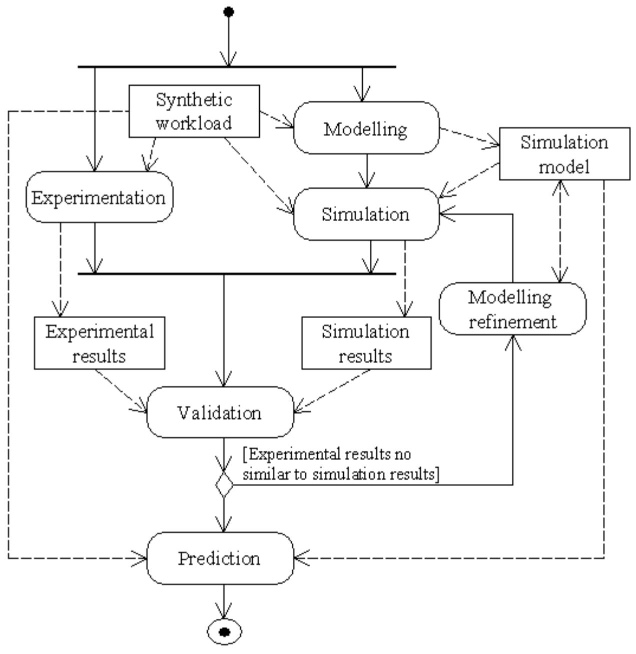

Figure 1. The simulation process.

Although the simulation process of a target system is very similar to a typical and generic simulation process, their activities are different. For instance, the `Modelling`, `Experimentation`, and `Prediction` activities are performed based on a synthetic workload of the target system. In fact, the simulation model produced in the `Modelling` activity and the experiment scenarios used in the `Experimentation` activity are derived from the synthetic workload. Moreover, the synthetic workload is used as it is in the `Prediction` activity.

3 The Case Study

The case study is described as follows. First, the migration context that helps the comprehension of the complexity of the legacy system, the target system, and the migration process itself, is presented. Then, the synthetic workload produced from the legacy and target systems according to the techniques presented in [3] is discussed.

3.1 Migration Context

CAPPLES was used to predict the behaviour of a target system, called SICOM, which was developed to replace the commercial system operating at COPASA-MG since 1985. This commercial system, called HP07, was a mainframe-based legacy system composed of about 3,500 programs written in COBOL and Natural. As a legacy system, HP07 was not fulfilling the organisation's needs. Thus, a migration effort was initiated to develop a completely new system, the SICOM.

After five years of development, SICOM was ready to replace the HP07. However, as this was a mission-critical system for COPASA-MG, its replacement should be carefully carried out. In fact, HP07 system was mission-critical system since was in charge of all commercial tasks and some of the operational tasks of an organisation responsible for water supply for more than 5 million customers. As the organisation was fully adapted to work according to the mean response times (MRTs) of HP07, CAPPLES was used to predict the MRTs of SICOM's on-line transactions. Therefore, the expected response times of the target system were known before the migration took place, so that the organisation could, if needed, adjust its services to work according to the predicted response times.

SICOM has 1,291,087 lines of code, divided into about 6,500 programs, and was entirely developed in Natural using the ADABAS database management system (DBMS) [13].In contrast with HP07, which processed data from the whole state of Minas Gerais in a centralised mainframe, SICOM was designed to operate in a distributed environment in seven distinct regions of Minas Gerais. In this study, we have analysed the behaviour of SICOM in the Northern Region's environment. However, the results of this work can be extended to the other regions in a simple manner. The Northern Region is divided in three districts named Montes Claros, Janaúba, and Januária. Each district has a local area network (LAN). The SICOM database server will operate in Montes Claros, and users from Montes Claros, Janaúba and Januária will use the system through `telnet` sessions. The LANs of these three districts are interconnected using X.25 links. This set of interconnected LANs specify a wide area network (WAN). Figure 2 illustrates this operational environment.

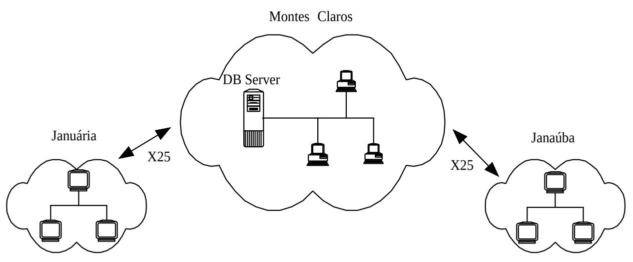

Figure 2. The Northern Region environment.

3.2 The Synthetic Workload

To understand how a synthetic workload can be composed, the following basic concepts are required:

- *On-line service.* An application can be modelled as a hierarchy of tasks, each one with a specific goal [8]. Tasks can be decomposed into subtasks, that can be decomposed into subsubtasks, and so on. Thus, tasks can be specified at different levels of abstraction. *Services* are top-level tasks composed of a set of conventional actions performed by their subtasks over the system. An *on-line service* is a service that requires interaction with users in order to achieve their goal.

- *On-line transaction.* There are subtasks that represent each interaction of the user with the application through the use of interactive devices, such as display, keyboards, and mice. These subtasks are *on-line transactions*.

- *On-line service's use case.* Usually, there are many ways to carry out an on-line service, each one combining a different set of on-line transactions. An *on-line service's use case* is one of the possible combinations of a set of on-line transactions required to perform an on-line service.

Considering these concepts, we can present a synthetic workload for the target system. Relevant on-line transactions are identified from measurements of the legacy system workload. The relevance criteria for selecting measured on-line transactions is presented in [1, 3]. From the relevant on-line transactions it is identified their on-line services, as presented in Table 1.

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 1. Identified on-line services.

The common decisions that are taken to perform an on-line service are identified and described in terms on-line service's use cases. These decisions can be even the cancellation of an on-line service. Having a well-established set of use cases, it is possible to individually measure the simulation parameters for each use case. Consolidating the measured parameters for each on-line service's use case we can produce a synthetic workload based on on-line services, as partially presented in Table 2.

Parameter	so-entry	so-finish	...	cust-upd
Intensity(%)	24	16	...	6
Number of interactions	5	3	...	2
Number of DBMS calls	115	86	...	33
Memory demand (MBytes)	NR	NR	...	NR
CPU demand (ms)	2070	1950	...	880
I/O demand (operations)	262	153	...	61
LAN demand (packages)	222	197	...	108
WAN demand (frames)	4290	3231	...	1692

Table 2. Partial view of the synthetic workload.

4 Building a Simulation Model

To model the computing environment of the target system, we used the SES/workbench [10] as our simulator. A discussion of the simulation techniques implemented in SES/workbench to provide its modelling constructors and to run its models, however, is out of the scope of this paper. In fact, a major aim of this paper is to provide a description of the model construction process. Further, this description should be provided in an abstract way in order to make it relevant for people involved in similar studies based on other simulators.

In SES/workbench, the computing environment of the target system is modelled as an ordered graph. *Nodes* of this graph are components of the computing environment. *Arcs* of this graph are binary relationships between components. *Simulation transactions* are processes that navigate through the arcs allocating, using, sharing, and releasing components. Nodes can be specified at different levels of abstraction. Concrete nodes represent the use of physical devices, such as storage and processing subsystems. Abstract nodes, or sub-models, are also ordered graphs that can be composed of concrete nodes and other abstract nodes. Considering these constructors, we can build the simulation model based on the synthetic workload.

One of the reasons for using SES/workbench is that it provides templates for most of the sub-models of our case study. The simulation model, as described in terms of SES/workbench constructors, is composed of 21 diagrams and has been presented in [1].

4.1 Building a Top-Level Model

The model should describe the computing environment from the point of view of the simulation transactions. For instance, the model should describe where the transactions are generated, processed and finished. Before explaining how the case study simulation model has been constructed, however, we would like to introduce three guidelines that should be considered when building simulation models using the CAPPLES approach.

Guideline 1 *The simulation model should be as simple as possible.*

Guideline 1 emphasises the fact that a simple simulation model may be so effective as a very detailed model. However, a detailed model probably would require more time and effort to be achieved than a simple model. Further, the validation effort of a detailed model also tends to be more complex than the validation effort of a simple model.

Guideline 2 *The construction of a simulation model should be an incremental and top-down process.*

Guideline 3 *Only resources used by the workload transactions should be modelled.*

Guidelines 2 and 3 emphasise that during the modelling process it may be the case that modellers can:

- lose the visibility of the entire computing environment;
- be influenced by ready-to-use templates and technological matters that may model more details than are required to produce a valid model.

Considering the guidelines above, we can explain the construction of the simulation model. Figure 3 shows a schematic representation of the top-level diagram of the computing environment used in our case study. The top-level diagram is composed of two abstract nodes, represented as boxes, that are sub-models of the simulation model. The role of each sub-model of the top-level diagram is explained as follows:

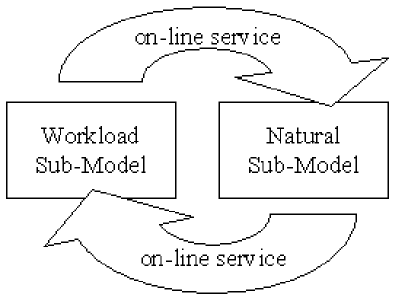

Figure 3. A schematic view of the top-level simulation model.

- *Workload Sub-Model.* Simulation transactions are created and destroyed in this node. Simulation transactions initially model on-line services, as presented in Table 1, that are submitted to the `Natural Sub-Model`. Once the on-line services are processed in the `Natural Sub-model`, they return to the `Workload Sub-Model` to be destroyed. The setting of the required parameters to process these simulation transactions is performed in this node, during the creation of the simulation transactions. These parameters are composed of data presented in Tables 1 and 2, as well with values used to express how on-line services can be decomposed in on-line transactions and database transactions.
- *Natural Sub-Model.* Simulation transactions modelling on-line services are processed by this node. Figure 4 shows a schematic representation of this abstract node. Boxes are also abstract nodes. The “clock” is a concrete node representing a delay on the execution of the simulation transactions. The hexagons are branches that take decisions based on current parameters of the simulation transactions. The meaning of each one of these abstract and concrete nodes in Figure 4 is explained as follows.
  - *On-line service decomposer.* The decomposition of on-line services into on-line transactions is controlled in this node. The `number of interactions` parameter in Table 2 is used by this node.

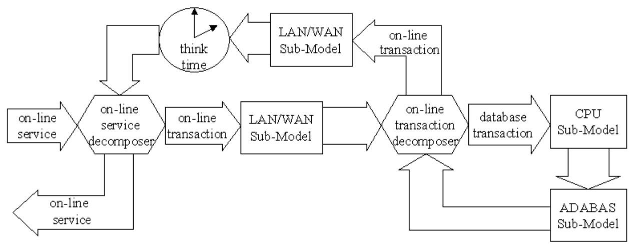

Figure 4. A schematic view of the Natural Sub-Model.

- *On-line transaction decomposer.* The decomposition of on-line transactions into database transactions is controlled in this node. The `number of DBMS calls` parameter in Table 2 is used in this node.
- *LAN/WAN Sub-model.* The use of the local area networks and remote links by each on-line transaction is modelled by this abstract node.
- *CPU Sub-Model.* In the `Natural Sub-Model`, this node models the amount of time the `CPU Sub-Model` has been used by the Natural process to perform an on-line transaction.
- *ADABAS Sub-Model.* The use of the ADABAS DBMS by each database transaction is modelled by this abstract node.
- `Think time.` This concrete node models users' *think time* (TT) [11]. In fact, TT is a period of time that users need to assimilate the information provided by user interfaces, to take a decision, and to implement such decision. TT is widely considered in performance analysis since it affects the performance of transactional workload [9, 7].

Each one of the abstract nodes in Figure 4 can also be composed of other abstract nodes. For this reason, the modelling can be a complex process that should follow Guideline 1. Exemplifying this complexity, the `ADABAS Sub-Model`, already introduced in Figure 4, is also composed of the following abstract nodes:

- *I/O Sub-Model.* The use of the storage subsystem by each database transaction submitted to the ADABAS Sub-Model is modelled by this node. In this case, statistics about the effectiveness of the buffer pool in the legacy system may be required to properly model the use of the I/O Sub-Model by the ADABAS Sub-Model.
- *CPU Sub-Model.* In the `ADABAS Sub-Model`, this node models the amount of time the `CPU Sub-Model` has been used by the ADABAS DBMS process to perform the database transaction's queries. This `CPU Sub-Model` is the same one used by the `Natural Sub-Model`.

Furthermore, abstract nodes can themselves be complex. For instance, the `ADABAS Sub-Model` should model logical locks that can happen when concurrent tasks are updating the same data. Recalling Guideline 3, data in the database that cannot be locked by the on-line services in Table 2 does not need to be modelled.

4.2 Modelling Physical Resources

A physical resource can be modelled by a concrete node that behaves like a queue. However, it is also a good approach to model physical resources using abstract nodes. For instance, the `CPU Sub-Model` of the database server (see Figure 2) could be represented by a concrete node modelling a single processor, as it is in the case study. However, it could be an array composed of two or more processors. In this case, rather than a concrete node, we can model the processor as an abstract node, that is, the `CPU Sub-Model`, with a concrete node modelled within it. A brief description of how the physical resources were modelled is given as follows.

- *LAN/WAN Sub-Model.* The use of LANs and remote links by on-line transactions is described by this node. A clear identification of the origin of the on-line transactions is important to select which LANs and remote links should be used. The `LAN demand` and `WAN demand` parameters in Table 2 is used within this sub-model.
- *CPU Sub-Model.* This is a simple processor representing the CPU of the database server. The `CPU demand` parameter in Table 2 is used within this sub-model. An important detail to be modelled in this sub-model is the overhead due to the database server's operating system. This overhead can be measured during the `Experiment` activity, described in Section 5.
- *I/O Sub-Model.* This node is modelled according to the specifications of the storage subsystem's maker specifications. The `I/O demand` parameter in Table 2 is used within this sub-model.
- *Memory Sub-Model.* Considerations about the technology in use in the target system computing environment has demonstrated that the required amount of memory will be constant. Further, it has demonstrated that the required amount of memory will be below the

available amount of memory. Recalling Guideline 3, we are not modelling this physical resource.

5 Generating Experimental Results

The `experimentation` has demonstrated to be the most time-consuming activity in the simulation process. The experiments should be repeatable to make their results useful to validate the simulation model. Further, user behaviour should also be repeatable since CAPPLES is based on on-line transactions. In this section we present the *virtual user*, *environment isolation*, and *environment customisation* techniques used in the experiments. Additionally, we present a strategy to create experimental scenarios.

5.1 Virtual Users

User behaviour, in this study, is characterised by the period of time a user needs to take and to implement decisions while interacting with an on-line transaction. This period of time is the TT, already introduced in Section 4.

Users may have a typical behaviour when interacting with computers, but they probably do not have a constant and repeatable behaviour. Automating user interactions during the execution on an on-line transaction is a suitable approach to make user behaviour repeatable. As the SICOM is based on TCP/IP telnet services [14], we have modified a telnet program to make it behave like a *virtual user*. Moreover, a set of virtual users were developed in order to implement some of the on-line services in Table 1.

Even though, it has not be easy to verify the accuracy of the virtual users. The following problems have been identified and solved:

- Timestamps should be recorded in memory during the measuring of the response time. The timestamp log can be dumped to a disk later, at the end of the experiment. In fact, the time required to record the timestamps to disk may be longer than the period of time between two consecutive timestamps. Thus, recorded timestamps could become meaningless.

- A virtual user should be implemented as a single process. Otherwise, it may be difficult to identify how the scheduling of the client computer's operating systems can affect the execution of more than one process. Once again, recorded timestamps could become meaningless.

The implementation of virtual users does not address all the requirements to have repeatable experiments. In fact, distributed environments have also contributed to make the experiments complex.

5.2 Environment Isolation and Customisation

The experimental computing environment must be isolated so that it is not be affected by elements not specified in the simulation model. For instance, using a local network that is shared with other computers not described in the model may affect the measured number of packages collected during an experiment. Moreover, software may require customisation in order to make them behave like in the real operational environment of the target system. For instance, database pre-fetching and buffering mechanisms should be customised in a way so as to not provide many benefits for processing the database queries. The following components of the computing environment have been identified to be isolated or customised:

- *Local area networks (LAN)*. Only computers used in the experiments should be connected to the LANs used in the experiments. Interconnected LANs that are not described in the simulation model should be disconnected from the LANs that belong to the experimental environment.

- *Remote links*. Like LANs, remote links that are not described in the simulation model should be disconnected from the WAN that belong to the experimental environment.

- *DBMS buffer pools.* The worst scenario in terms of performance is usually the best to indicate limitations of the computing environment. DBMS buffer pools can minimise the DBMS performance since they the submitted queries can be affected by some data locality (the principle of locality [6]). Therefore, the idea is to customise the DBMS buffer pool to reduce its efficiency as much as possible.

- *Natural buffer pool.* Like the DBMS, the Natural environment also has a buffer pool that may reduce the loading time of Natural programs [12]. This buffer pool should then also be unconsidered in the experiment.

5.3 Experimental Scenarios and Results

Using virtual users in an isolated environment has demonstrated to be essential to make the experiments repeatable. Indeed, the experimental results used to verify the accuracy of the experiment techniques are always exactly the same for a specific virtual user running in the isolated and customised environment. Then, it has been possible to measure the MRT of the on-line transactions in the scenarios described in Table 3.

Scenario number	Number or concurrent virtual users	Virtual user's on-line service
1	3	so-entry
	2	so-query
	3	cust-rpt
2	5	so-entry
	3	so-query
	1	cust-rpt
	5	cust-info
3	8	so-entry
	5	so-query
	2	cust-rpt
	9	cust-info

Table 3. Experimental scenarios.

The simulation scenarios were created from on-line services of the synthetic workload. On-line services that update the database in a non-repeatable way have not been considered in the experiments. This, however, does not mean that there are no services that update the database. For instance, the `so-entry` service in Table 3 updates the database.

Virtual users have also been essential to produce relevant experimental results using a computing environment smaller than the computing environment of the target system. For instance, a transactional workload is inversely proportional to the TTs [7, 9]. Thus, having small TTs, as those that can be produced by virtual users, it is possible to generate a transactional workload even bigger than the expected real workload. For instance, we have used a TT of 0.5s in the experiments, 1/60 of the 30s TT used to make the predictions.

Table 4 presents the experimental results produced from the consolidation of the measured timestamps. The major difficulty of the implementation of the experimental scenarios has been the allocation and customisation of the software and hardware, as described in Section 5.2. Even though the implementation of the three scenarios in Table 3 has shown that the measured MRTs in Table 4 are not linear. Thus, the scenarios are complex enough to produce workloads that are effected by the concurrency of on-line transactions competing for shared resources.

Scenario	Mean Response Time (ms)				
number	so-entry	so-query	cust-rpt	cust-info	Average
1	231	271	386	–	299.12
2	296	241	852	234	478.35
3	446	358	1,537	328	826.96

Table 4. Experimental results.

6 Simulation Model Validation

The execution of the model is the first step to validate the model. The parameters used to execute the model in the simulation study were the same for validating the model, and for producing the predicted results. The simulation parameters are presented as follows:

- *Observation period.* The millisecond (ms) is the simulation unit. Being the observation period 1 hour long, it has 3,600,000 simulation units.
- *Warm up period.* The same of the observation period: 3,600,000 simulation units.
- *Confident interval specification.* 95% of precision for building a confident interval based on the MRT of on-line transactions.

The simulation model has undergone three major refinements in this study. However, a thoughtful discussion of how the model has been refined would require a more detailed explanation of the simulation model than that presented in Section 4. Further, a complete understanding of the SES/workbench constructors would be required to understand the SES/workbench-based model[1]. Even though, we introduce two guidelines that summarise the lessons we have learned.

Guideline 4 *Avoid the use of different time granularities throughout the model.*

Guideline 4 should be considered when using predefined templates for constructing the simulation model. For instance, the *ethernet* and X.25 templates can be used for modelling LANs and WANs based on these technologies. Thus, these templates can be used to identify performance bottlenecks in some LANs and WANs. However, the default time granularity of an ethernet template should be smaller than the default time granularity of a X.25 template. Increasing the time granularity of the ethernet template, however, may make the ethernet template useless.

Guideline 5 *Concurrency effects are mainly caught by logical resources.*

Guideline 5 indicates that logical resources can be modified more easily than physical resources and workloads. In fact, physical resources and workloads usually have well-known parameters that can make them inflexible.

The validated simulation model can produce *results no more than 5% different of the results produced for each experimental scenario*. Indeed, this is the criteria that we have used to consider the simulation model valid. For instance, the simulation results cannot meet this criteria and be linear at the same time. In fact, this difference could even be higher than 5% to guarantee no linear results.

[1]A comprehensive description of the SES/workbench-based model is presented in [1].

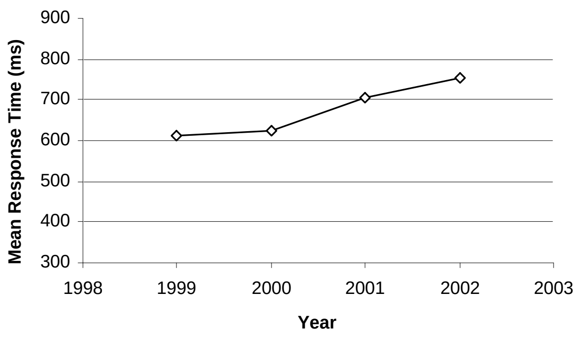

Figure 5. Predicted MRT of the target system's on-line transactions.

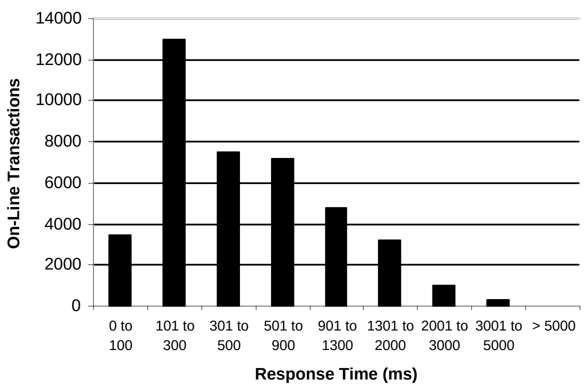

Figure 6. Predicted histogram of the target system's on-line transaction response times.

7 Predicted Behaviour

Once we have a valid simulation model, we can start the `Prediction` activity. To perform this activity, we may use the *business unit* concept presented in [9]. The business unit in our case study, for example, is the *number of customers*. At COPASA-MG, the Commercial Department is the organisation's unit that effectively uses the on-line services in Table 1. Thus it had statistics demonstrating that its usage of these on-line services is proportional to the number of customers. Moreover, the COPASA-MG has historical statistics of the growing rates of customers per year, region and council. Therefore, it has been possible to apply these customer growth rates on the intensity of each on-line service of Table 2, producing the graph shown in Figure 5, predicting the MRT of the target system. Indeed, the results in Figure 5 are the most important results in terms of performance analysis to identify if the target system is going to suit or not into the organisation requirement. In the SICOM case, we have predicted that its performance would not be a problem. Indeed, the MRT of the on-line transactions on the legacy system was approximately 900s, almost 300ms longer than the predicted 612.071ms (confidence interval: [587.235ms, 636.906ms]) for the target system.

However, the average MRT alone is not enough to evaluate the performance of a target system. In fact, it may be possible that a small, but relevant percentage of on-line transactions may have unacceptably long MRTs. Having a validated simulation model we can also produce a distribution of the MRTs, as shown in Figure 6. In our study, we can see that no transactions has a MRT of more than 5s, confirming that the migration to the target system should be smooth in terms of its performance.

Moreover, a validated simulation model can also be used for capacity planning. Table 5 presents the utilisation of the modelled resources of our case study. This utilisation table was produced along with the prediction of MRT for the year 1999. From Table 5 we can see that LANs are not a concern in terms of performance. Moreover, the database server's CPU should be the physical device with highest utilisation. Even so, the database server's CPU is far to be considered as a potential bottleneck with only 48.184% of utilisation.

Resource	Utilisation (%)
Database server's CPU	48.184
Database server's I/O subsystem	1.557
Montes Claros local network	0.071
Januária's local network	0.011
Janaúba's local network	0.007
Remote link between Montes Claros and Januária	20.347
Remote link between Montes Claros and Janaúba	18.882

Table 5. Resources utilisation.

8 Conclusions

This paper presented a complete description of a simulation process used to predict the behaviour of a real target system before it became operational. Although the simulation process presented in this paper has many similarities to typical simulation processes, there are some differences concerning the legacy system migration context that should be pointed out:

- The modelling, experimentation and prediction activities in the simulation process are based on a synthetic workload created from information collected on the legacy and target systems.

- On-line services identified during the characterisation of a synthetic workload in CAPPLES are used:
  - to guide the building of the simulation model, identifying the computing environment elements that should be modelled;
  - to generate experimental scenarios.
- The on-line transaction's use case technique used to characterise synthetic workloads in CAPPLES is used to implement virtual users that are essential to make the required experiments repeatable.

Therefore, the simulation process for performance evaluation of a target system requires specialised activities rather than typical simulation process activities.

Moreover, the case study presented in this paper has indicated that the adopted simulation strategy may be useful for similar studies. Indeed, the presented simulation strategy is an integrated part of the generic CAPPLES method that is, to the best of our knowledge the unique method that explains how to carry out a performance evaluation during the migration of a the legacy system.

Acknowledgements. The first author is sponsored by Conselho Nacional de Desenvolvimento Científico e Tecnológico - CNPq (Brazil) – Grant 200153/98-6.

References

[1] 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).

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

[3] P. Pinheiro da Silva, A.H.F. Laender, R. S. F. Resende, and P. B. Golgher. Characterizing a Synthetic Workload for Performance Evaluation during the Migration of a Legacy System. In *Proceedings of the 4th European Conference on Software Maintenance and Reengineering*, pages 173–181, Zurich, February 2000.

[4] D. Ferrari, G. Serazzi, and A. Zeigner. *Measurement and Tunning of Computer Systems*. Prentice Hall, Englewood Cliffs, NJ, 1983.

[5] J. Gunther. Capacity planning for new applications throughout a typical development life cycle. In *Proceedings of the Computer Measurement Group Conference*, Nashville, TN, December 1991.

[6] J. Hennessy and D. Patterson. *Computer Architecture: A Quantitative Approach*. Morgan Kaufmann, San Francisco, CA, 1996.

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

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

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

[10] Scientific and Engineering Software, Austin, TX. *SES/*Workbench, *Creating Models, Release 3.1*, 1996.

[11] B. Shneiderman. *Designing the User Interface: Strategies for Effective Human-Computer Interaction*. Addison-Wesley, Reading, MA, 1992.

[12] Software AG, Darmstadt, Germany. *Natural Installation and Operations Manual for UNIX*, June 1995.

[13] Software AG, Darmstadt, Germany. *ADABAS Concepts and Facilities Manual, Version 6.2*, August 1997.

[14] W. Stevens. *TCP/IP Illustrated, Volume 1: The Protocols*. Addison-Wesley, Reading, MA, 1994.

[15] C. Wilson. Performance engineering-better bred than dead. In *Proceedings of the Computer Measurement Group Conference*, pages 464–470, Nashville, TN, December 1991.