Towards the use of Qualitative Reasoning for supporting Information Technology Management

Ricardo M. P. de Alcântara and **Germana M. da Nóbrega**
Mestrado em Gestão do Conhecimento e da Tecnologia da Informação – Universidade Católica de Brasília (Brazil)
e-mail: piquet@tba.com.br, gmnobrega@pos.ucb.br

Paulo Salles
Instituto de Ciências Biológicas – Universidade de Brasília (Brazil)
e-mail: paulo.bretas@uol.com.br

Abstract

Building up human resources for Information Technology management and improving web services would benefit if causality was taken into account. In fact, operation manuals and handbooks seldom explore causal relations involving components like Web Servers, Application Servers and Database Servers in tasks such as capturing messages, monitoring and capacity planning. Indeed, causality is sometimes only implicit in documents or in the everyday practice of companies. In this paper we discuss the use of qualitative reasoning techniques for managing IT using an implemented simulation model as illustration. We argue that having explicit representations of objects, configurations and causal relations typically found in qualitative models may be of great importance for understanding IT systems and useful for training support and operation teams.

Introduction

As a crucial part of business activities, managing IT services is often a complex task for financial banks. Requirements on this kind of service are always broad and problems may cause loss of customers and money. IT services include a variety of components. For example, in order to provide costumers means for paying bills, consulting balances or making investments a set of Web transactions is required. Web components involved include Web Servers, Application Servers, Network, and Database Servers. Monitoring and control are necessary to keep up with the quality of services provided by the financial banks.

Currently, operation and support teams should monitor messages coming from each of the components and integrate them. Such information is managed without any reference to related events, limiting thus the effectiveness of environment management. A number of initiatives have been developed to overcome such difficulties. One of such initiatives is the IT Infrastructure Library - ITIL[1], which offers documents to be used in the implementation of a framework for IT Service Management (van Bon 2005). ITIL includes two important titles, service delivery and service support. Service delivery includes several disciplines like problem management and incident management. Service support includes other disciplines like capacity management and availability management.

The dynamics engendered for delivering services to customers seems to establish a causal chain between its components. As many components are simultaneously affected, we assume that such a chain, if rendered explicit, might help managers and support teams to understand and solve problems. Using traditional approaches in this area, such as clustering messages, has proven to be of difficult implementation due to the huge number of messages and informations available to the decisions makers. In this context new approaches have to be considered. In this paper we propose to address some disciplines from ITIL by using Qualitative Reasoning (QR) techniques (Weld & de Kleer 1990). Particularly, we focus on capacity management and incident management, because these two areas include investigation and diagnosis tasks and performance management. The main motivation for exploring QR in this new area comes from successful stories in different contexts in which understanding of the system behaviour is grounded in a description of system structure, that is, of components and relations between them. This understanding enable different applications, for example, diagnosis (Heller 2001) and education (Salles & Bredeweg 2003).

This paper is organized as follows: in section 2, we present the context for this study and details of the problem. We justify the use of a qualitative approach to IT problems in section 3 and in section 4 we describe a qualitative simulation model. In section 5 we present some simulations and the results are discussed in section 6. Finally in section 7 we present our conclusions and mention ongoing work.

Incident Management in Web Environment Infrastructure

Customers using a Web browser execute transactions in Web pages. These pages are requested to a Web Server, which sends pages to the customer. The Web Server requests services to an Application Server through an internal Network. Business programs in the Application Server need data to process and, in turn, request services from the Database Server, which gets data either from buffer memory or disks and deliver them back to the Application Server.

[1] http://www.itil.org.uk/

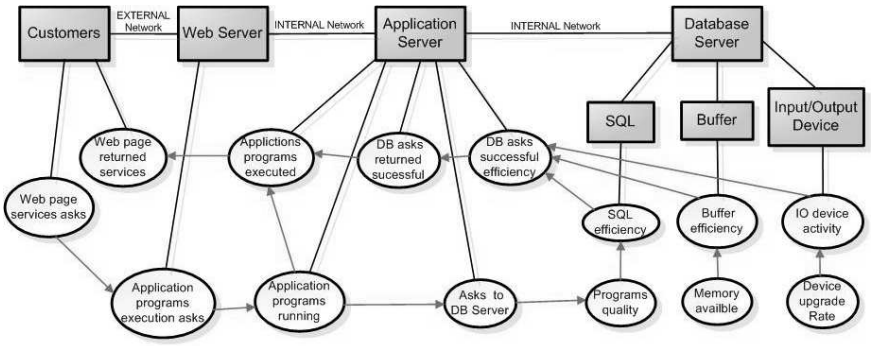

Figure 1: Simplified representation of components involved in Web transactions.

As it occurs, for instance, in a financial bank, the information flow of the Web transactions is depicted in Figure 1 which shows the dependence among the components (Armstrong *et al.* 2005, Chapter 1: $Overview$, Section $Container$, Subsection *Container Services*). A business program can either access or not the Database Server. If not, the program returns returns its results to the Web Server, which in turn sends the new page to the customer's browser. If a Database Server has to be accessed, the control is passed through, and the application program waits for an answer. We identify casual chains among the following internal parts of database components:

- SQL - this part is responsible for interpreting SQL statements and to determine the amount of data to be analyzed;
- Buffer - after a SQL statement is interpreted, the database manager looks into buffer to get data;
- IO device - if data is not found in buffer, they should be searched in the IO devices; getting data from the IO device is slower than getting them from buffers.

In each of these internal parts we identify a process: program quality influencing the SQL efficiency, memory available influencing the buffer efficiency, and the rate of upgraded IO devices influencing the IO device wait. These three processes influence the database behaviour (Sanders 2003), which, by their turn, influence the behaviour of the application server and are depicted in Figure 1. These process are further investigated below.

The database performance relies on the basic principle that retrieving data from memory is better than retrieval from disk (IO devices), since this activities are often more time consuming, and are likely to cause delay in the whole process (IO wait). It is a widely accepted rule that IO wait should be avoid. Whenever IO waits are pending, application programs remain in waiting state, until they receive a successful response from the Database Server and may continue to run.

The most important process for the database performance is the program quality influencing the SQL efficiency. SQL efficiency depends on the balance between good and bad programs. The quality of the program code is assessed by the amount of rows that a database analyses in order to retrieve a result. A good program requires a smaller number of rows to retrieve data than a bad one. Often programmers produce bad programs when they mis-encode the SQL statement. In this case, it is likely that many of the retrieved rows will be discarded until the query result is returned.

Table 1: Illustrating customer table.

ID	Name	Address
0655567767	João	STN 716
2323909656	Pedro	SQN 603
....	....	.....

Consider the following example involving a bad program. Let Table 1 represent a customer table, described by the columns `id`, `name` and `address`.
Suppose this table is indexed by the `id` column. An application programmer writes the following SQL statement:

```
Select id, name, address where name = Pedro
```

Since the table is indexed by the `id`, the database manager system (DBMS) in Database Server will read all the existing rows from the table in order to select those having "Pedro" as a `name`. As the whole table will be searched probably the data are not in buffer memory and the disk has to be searched to retrieve the required information. SQL efficiency can be calculated from the amount of data searched by good and bad programs. If the former is greater than the latter, the Database Server has good or excellent efficiency, otherwise has bad or very bad efficiency.

In addition to the program quality influencing the SQL efficiency, two other relevant processes for the database performance have the following effects:

1. changes in the database buffer memory availability: increasing memory may improve buffer efficiency;
2. Technological updating IO devices may change the system's IO wait: incorporation of technological inovations may reduce IO wait.

Noticed that these two processes result from actions originated outside the database system. In fact they are not under the responsibility of database support team.

IT infrastructure problems have to be tackled before they reach other system components. In the context of propagation of problems, it is not surprising that sometimes an event

is misinterpreted by a human operator and that actual problems are only detected after a phone call from a customer to the help-desk.

Another relevant aspect of Web transactions management and IT management in general is the difficulty for training people that are able to deal with so many different components. Ranging from general to more specific components, the support team includes many person-levels for monitoring and solving problems. In general component-specific experts acquire problem solving skills by hands on training, because much of such knowledge cannot be found in manuals or textbooks, or sometimes it is spread over a number of sources. It may happen also that IT management knowledge is implicit and related to the specific company environment.

From the description above, a remarkable feature of IT management is that much of the knowledge used in managerial activities is heuristic and based on (expert) commonsense. In fact, although manuals present details about the functioning of the components and procedures to check them up, at a higher level of organization in a company often require problem solving skills aquired by hands on training. It is also remarkable the fact that IT support workers often have to decide using incomplete knowledge, initially presented as verbal descriptions of the problems. Finally, it is possible to identify causal relations between the elements found in a typical setting of IT administration. This situation can be illustrated by relation involving the internal parts of the Database Server component and its relationship with an application server. Software companies offer a number of manuals to describe the behaviour of their products internal parts but do not describe how these parts influence each other. A great deal of information involved in software performance is discussed within user groups or in companies Web sites (Sanders 2003).

IT services can be further improved by the addition of new approaches to the traditional techniques of capturing messages and taking actions without a general view of the business services and processes involved. In the next section we justify the use of a Qualitative Reasoning approach for modelling these problems.

A Qualitative Modelling Approach

Qualitative Reasoning (QR) is an area of Artificial Intelligence that deals with representations of continuous aspects of the world to support reasoning with little information. A number of techniques and modelling paradigms have been developed for QR (Weld & de Kleer 1990) and an updated overview of the achievements of the area can be found in (Bredeweg & Struss 2003). In this section we discuss aspects of the representational apparatus for QR modelling, particularly the one developed for this work, the Qualitative Process theory - QPT (Forbus 1984).

Quantity values are used in QPT to represent possible qualitative states a quantity can be found. Qualitative values are actually combinations of two values, magnitude and derivative. The former represents the 'amount' of stuff, and the latter, the direction of change. Possible values of magnitudes are, for instance, $small$, $large$, $normal$; for derivatives, the standard values are positive, zero and negative, meaning the variable is increasing, stable and decreasing. The set of possible of qualitative values of quantities is represented as a set of possible qualitative states the quantities may assume, called $quantity\ space$ (see e.g. (Forbus 1984)). For example, the entity $bank\ account$ can be associated to the quantity $balance$. Magnitudes of this quantity may draw on the quantity space QS = $zero$, $small$, $average$, $large$ where $zero$ and $average$ are points that represent absence of money or, lets say, the yearly average amount of money found in that bank account, and $small$ and $large$ are intervals with infinite possible numerical values below and above the average balance.

Two modelling primitives are of special interest: *direct influences*, posed by processes, that adds or removes stuff from the directly influenced quantities, and *qualitative proportionalities* or indirect influences, that propagate changes initiated by processes. Direct influences, modelled by I+ and I-, mean that the influencing quantity (a rate) is used to calculate the influenced quantity derivate value. For example, if I+(X,A) and this is the only direct influence on X, the derivative of X takes the value of the rate A. If the latter has a positive value, X increases. Similarly, if I-(X,B), this is the only influence on X and the rate B has a positive value, then B decreases by an amount equal to B's value.

Qualitative proportionalities, in turn, are modelled by P+ and P- and establish a relation between two quantities in a way that the influenced quantity gets the derivative sign of the influencing quantity. For instance, if P+(C,X) and this is the only indirect influence on C, this quantity will change in the same direction as X. Thus, if X is increasing, C will also increase. Similarly, if P-(D,X), this is the only influence on D and X is changing, then D will change in the opposite direction.

Direct influences and proportionalities have both mathematical and causal meanings. The former are qualitative representations of ordinary differential equations, where constraints are put on the derivative of a quantity. Qualitative proportionalities carry much less information: they represent some (maybe unknown) monotonic function that relates two quantities in a way that they either change in the same or in opposite directions. The causal meaning of direct influences and proportionalities is very clear: causality is directed and in both cases, I$\pm$(Q,R) and P$\pm$(S,Q), the second argument always influence the first one, never the contrary. This is how causal chains are built up: a direct influence changes the derivative of an influenced quantity and this change propagates to other quantities via qualitative proportionalities. For example, consider the following model:

I+(Balance,Deposit); I-(Balance,Debt);
P+(Manager satisfaction, Balance); P-(Financial costs, Balance)

This model allows for predictions such as "If the rate of deposits is greater than the debt rate, then the balance increases. If the balance is increasing, the manager satisfaction is increasing too, and the financial costs for the customer are decreasing". If the causal chain is examined in the opposite direction, the model supports explanations such as "The financial costs of bank operations are increasing and

the manager satisfaction is decreasing because the balance is decreasing. It is happening because the debts within this period are greater than deposits."

For this work we use Garp3 workbench for qualitative reasoning and modelling (Bredeweg *et al.* 2006). This new software combines the qualitative simulator Garp (Bredeweg 1992) and the related tools Homer (Machado & Bredeweg 2002) and VisiGarp (Bouwer & Bredeweg 2001). Garp3 allows for ontologies like QP theory and other to be used for building qualitative models, and has been used in a number of modelling activities. An interesting feature of qualitative models is compositionality, that is, the possibility of assembling partial models into more complex models. This work adopts the compositional modelling approach (Falkenhainer & Forbus 1991), so that model building is like building a library of *model fragments*, that can be combined into different simulation models about the same domain of knowledge.

Garp3 uses three types of model fragments (static, process and agent) to capture different types of knowledge. Simulations always start the description of an initial scenario from which the simulator select appropriate model fragments for dynamically build a model. From the knowledge available in the active model fragments, Garp computes the values for the quantities and create the possible states derived from the scenario. A qualitative $state$ can be defined by the qualitative values of the quantities, and lasts a certain time period. Garp looks for possible transitions and computes again the values of the quantities, given the conditions found in each state, in order to produce new states. Garp iteratively computes new values for the quantities and creates new states, until no more transitions are possible. The full picture of the simulation, the *behaviour graph*, consists of all possible states and state transitions, given the knowledge encoded in the library and the conditions of the system specified in the initial scenario. Each sequence of states is a *behaviour path* and normally ends in equilibrium states.

The next section presents the implementation of a qualitative model about the problems described in section 2 in Garp3 and section 5 presents some simulations with the model.

A Qualitative Model for IT Management

We describe in section 4.1 the proposed qualitative model by means of its building blocks (i.e., entities/agents, quantities and quantity spaces), providing an explanation of its purpose. Then, in section 4.2, we explore these building blocks to present some model fragments and a scenario.

Defining building blocks

The purposes of the model are to show how distinguished processes, managed by different teams, may influence the performance of a database responding to requests from Web programs and to provide support for the idea that being aware of the underlying causal chain might be relevant for supporting management of computational services.

The hierarchy of entities included in the model, as well as description for each entity are provided in Table 2.

Table 2: Entity summary.

Entity	Super type	Description
Servgbd	Entity	Database Server
Servappl Sql	Entity Servgbd	Application Server Optimizer- Sql Interpreter
Input Output device	Servgbd	DASD - direct access service disks, library tapes, etc.
Buffer	Servgbd	Memory area to store data and indexes from relational tables
Device upgrade	Agent	Machines upgraded or obsolete
Memory	Agent	Memory available for buffer activity

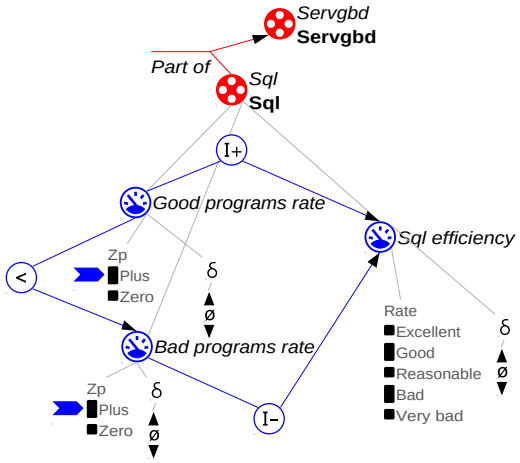

Figure 2: Process fragment showing the program code quality influencing SQL efficiency.

In addition, Table 3 shows quantities that we assign to those entities, along with quantity spaces and a description for each quantity.

Model fragments and a scenario

Figure 2 illustrates the process model fragment showing SQL efficiency influenced by the quality of program code: I-(SQL efficiency, bad programs rate), I+(SQL efficiency, good programs rate).

The inequality [good programs rate < bad programs rate] included in the model fragment means that SQL efficiency is under risk, because more bad programs are being produced by the development team for accessing the Database Server than good ones, and a large amount of data to be analyzed is expected. The static model fragment in Figure 3 shows that whenever SQL efficiency is low, the amount of data that should be searched is large: P- (SQL data, SQL efficiency). An inverse correspondence is included to relate the values of the two quantities. For example, an excellent efficiency corresponds to very few SQL data analyzed by the database.

The amount of searched data due to a SQL statement affects the buffer hits rate and the database efficiency for answering queries because as many data are unnecessarily analyzed to each statement, it is likely that they are not in buffer: P-(buffer efficiency, SQL data) and P- (DB efficiency, SQL

Table 3: Quantity summary.

Quantity	Entity/ Agent	Quantity space	Description
Bad programs rate	Sql	{zero, plus}	The speed with which bad programs require database services
Good programs rate	Sql	{zero, plus}	The speed with which good programs require database services
Sql efficiency	Sql	{very bad, bad, reasonable, good, excellent}	Assessed by the amount of rows that should be analyzed by the database to retrieve a result
Sql data	Sql	{very few, few, medium, lot, too much}	Amount of data analyze by the database to retrieve a result
Buffer efficiency	Buffer	{very bad, bad, reasonable, good excellent}	The number of times data are searched in the buffer and are found
Buffer returned data	Buffer	{zero, critical, medium, lot, paradise}	Amount of data searched in the buffer and successful returned
IO device activity	Input Output device	{very few, few, medium, lot, too much}	IO device activity needed to recover data not found in the buffer
IO device wait	Input Output device	{very few, few, medium, lot, too much}	Wait time of the system relative a IO process in disks
IO device upgrade	Device upgrade	{minus, zero, plus}	The rate of replacement old devices by new ones
IO device upgraded	Device upgrade	{few, average, many}	The number of devices upgraded or up-to-date
IO obsolete device	Device upgrade	{few, average, many}	The number of obsolete devices
Memory available	Memory	{minus, zero, plus}	The amount of avaiable memory
Db efficiency	Servappl	{very bad, bad, reasonable, good excellent}	Database efficiency in answering application

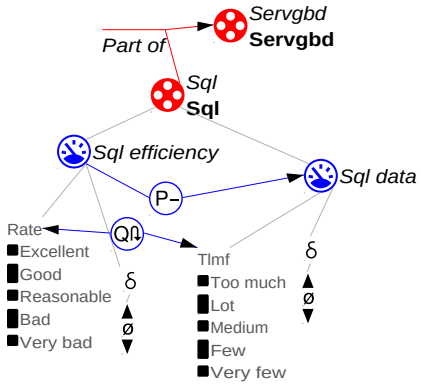

Figure 3: Static fragment relating SQL efficiency and the amount of retrieved data from queries.

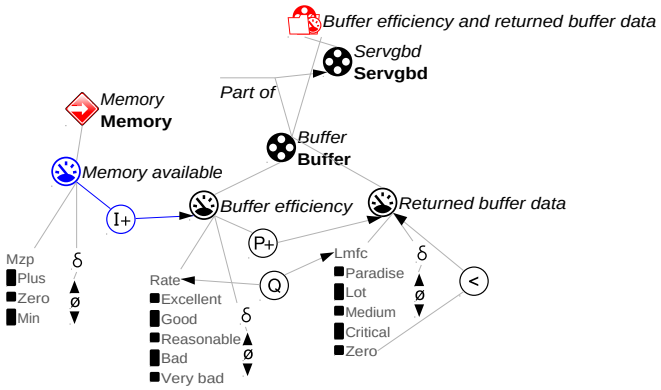

Figure 4: Agent fragment: a process changing buffer memory.

data). This happens because buffer usually keeps only data often accessed. As more data are read from disk, they are moved to the buffer. The buffer has a bounded memory to store data, so that as new data arrive, less recently required data are discarded from the buffer.

The above explanation leads us to the consequences of the process on other database internal services. The greater the number of times data is searched and found in buffer, the greater the amount of data returned from the buffer and the greater the efficiency of the database server for answering queries. These relatios are captured by the following proportionalities: P+(returned data buffer, buffer hits rate) and P+(DB efficiency, returned buffer data). In case that few data is found in buffer, it is necessary to find them in IO devices (disk): P-(IO device activity, returned buffer data). The greater the activity of the IO device, the greater the IO device wait and worse the database efficiency: P+(IO device wait, IO device activity) and P-(DB efficiency, IO device wait).

The amount of IO device wait affects the rate of both good and bad programs requests the database in a similar way: P-(good programs rate, IO device wait), P-(bad programs rate, IO device wait). This happens because IO device wait halt indirectly many database resources so that the system become overcharged and new applications have to wait more time to do database requests. This is a constraint of the model. If IO device wait variable reach a “too much” value, no more programs are to be accepted by the database, and if this situation lasts for a while the database will suffer an abnormal end, or the whole system halts.

In addition, the model includes two external processes that affect the database performance. These processes are often under the responsibility of different teams that usually are unaware of each other’s accomplishments.

We firstly create an agent allowing to simulate the process of changing buffer memory, considering that increasing memory for buffer may improve its efficiency (Figure 4).

Secondly, we create another agent allowing to simulate updating IO devices. In this case, we consider that technological updating IO devices may change the system’s IO device wait (Figure 5).

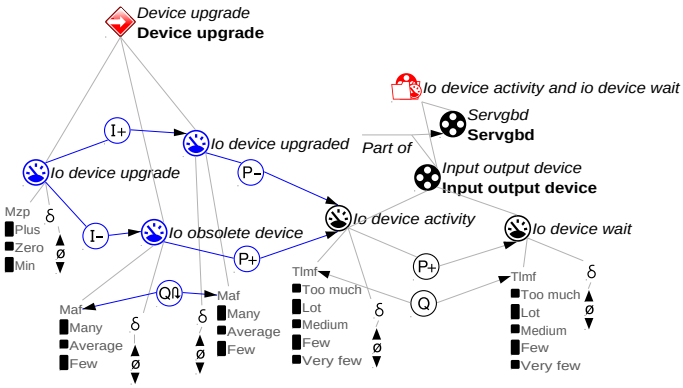

Figure 5: Agent fragment: a process of technological IO devices upgrade.

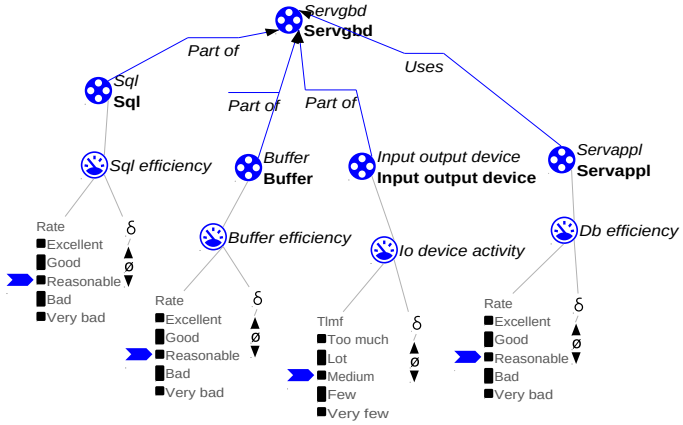

Figure 6: Scenario setting services at an intermediate level.

In order to create a scenario for simulating the system behaviour we suppose that new applications have been deployed and the relationship between good and bad programs is deliberately left unspecified so the simulation should demonstrate all three possibilities: [good programs < bad programs], [good programs = bad programs] and [good programs > bad programs]. The scenario defines that the remaining database services are working at an intermediate level (Figure 6).

Model Simulation

The database performance is strongly affected by decisions made during SQL statement interpretation and, as suggested before, by the amount of data that should be searched for each interpreted SQL statement. This is represented in the simulation deployed in section 5.1, pointing out to the (in)equalities between good and bad programs. The resulting causal model, following the compositional modelling approach (Falkenhainer & Forbus 1991), is shown in Figure 7.

Moreover, two external factors, represented by Garp3 agents introduced in section 4.2, may also affect the database performance (amount of available memory and IO devices

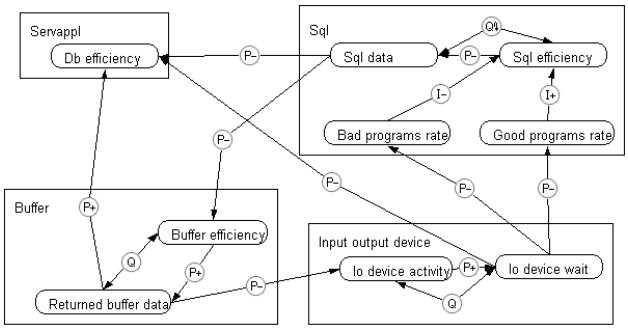

Figure 7: General view of the causal model in Garp3.

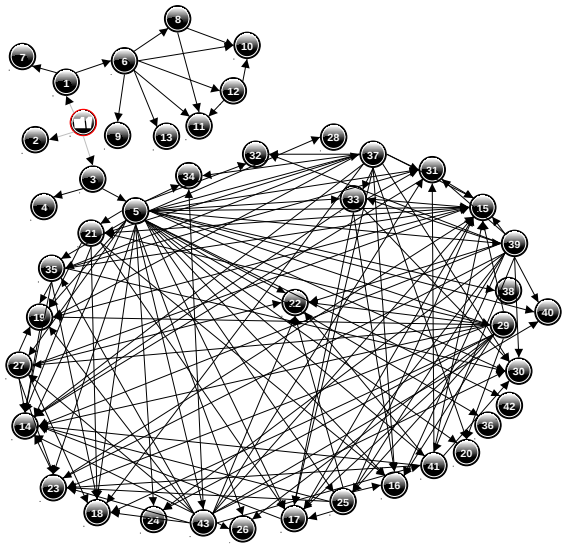

Figure 8: Full state graph for the SQL efficiency simulation.

technology). In section 5.2 and section 5.3 we discuss these changes whenever they occur during the simulation.

SQL efficiency simulation

Figure 8 contains the state graph of the full simulation resulting from the scenario introduced in section 4.2, considering the relationship between good and bad programs.

States 1, 2, and 3 represent possible behaviours assuming [good programs < bad programs], [good programs = bad programs] and [good programs > bad programs] respectively. The situation in state 2 shows the system in equilibrium: the system processes the same amount of SQL well-coded programs and bad-coded ones, allowing the database to work normally such as to answer to the queries under an acceptable buffer efficiency level, with the activity of IO devices at an intermediate level. We can say that data requested to the DBMS are in part responded by means of data already stored in the memory and in part by means of data retrieved from the disk.

The situation described in state 3 changes to the one described in state 5, and from here to a variety of behaviour paths. SQL efficiency improvement shown in states 3 and 5

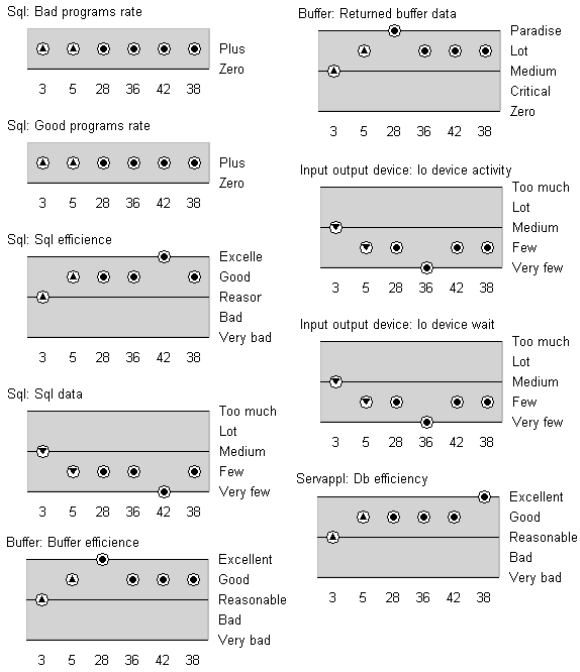

Figure 9: Value histories of behaviours path initiated in state 3.

has the following consequences:

1. decrease of the amount of data that should be searched (SQL data);
2. improvement of buffer efficiency (buffer efficiency);
3. decreasing of IO activity (IO device activity); and
4. improvement of efficiency in database response (db efficiency).

Value histories in Figure 9 show the transition from state 3 to state 5 and four equilibrium possibilities in states 28, 36, 42 and 38, derived from state 5.

The positive value (a triangle upwards) of the derivative of quantities good and bad programs comes from the negative proportionality engendered by the quantity IO device wait. This positive value suggests that when [good programs > bad programs], the database is capable of responding to more good and bad programs with good or excellent efficiency (measured by db efficiency).

Equilibrium states 28, 36, 42 and 38 are distinguished by the level of the systems's final state. Quantities *SQL efficiency*, *buffer efficiency*, *returned buffer data* and *db efficiency* have values ranging from $good$ to $excellent$ in these states. Quantity values of *SQL data*, *IO device activity* and *IO device data* range from few to $very\ few$.

Particularly, the analysis of the end state 28 shows that data buffer may reach an excellent performance due to SQL efficiency improvement. However, even when buffer has excellent efficiency level, the IO disk activity may yet be significative. Efficiency of database response (*db efficiency*) has an excellent level and the quantity for IO activity (*IO device activity*) has value few (but not *very few*). This state

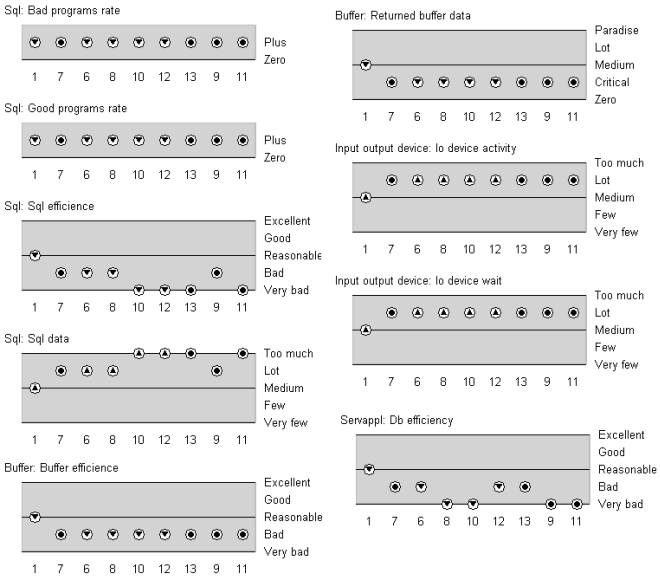

Figure 10: Value histories of path behaviours initiated in state 1.

suggests that IO activities, provided they occur at adequate levels, do not cause problems to the efficiency of database response, correctly corresponding to the reality. The functioning of the database may vary within several different levels of excellence without compromising the response of carried out requirements, i.e., if SQL efficiency is average or greater, the database performance may be acceptable for its invokers. In this favorable situation, the manager may administrate the database performance aiming to reach the maximum performance level, if this is a requirement for the company.

The situation described in state 1 [good programs < bad programs] accounts for less possible behaviors, since it is bounded by the following correspondence conditions:

1. if the IO level is too much [IO device wait = too much], the system becomes overcharged and new applications spend more time to do database requests. If this situation lasts for a while the database will suffer an abnormal end or the whole system halts;
2. even within very high levels of IO activity, some data is still retrieved from buffer. Thus, [returned data buffer > 0].

Changes in quantity values following state 1 are show in Figure 10. This figure also shows eight possible end states derived from state 1.

A possible behaviour path in this simulation can be described as follows:

- [1 → 7] - due to the arrival of more bad than good programs, SQL efficiency is decreased and ends up in a stability state with value bad, the same assumed by the quantity *buffer efficiency*. The IO activity variable is increased and changes its value to lot. Finally, database efficiency response (*db efficiency*) remains stable with value bad.

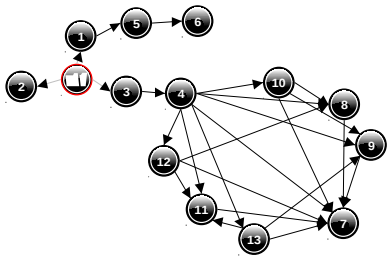

Figure 11: Simulation showing changes in the amount of memory due to the action of an external agent.

The analysis of this path points out to a degradation of database services and to equilibrium at a lower level. The negative proportionality engendered by the quantity *IO device wait* within the good and bad programs variables, lead them to a stable state at a lower level. This happens because the new IO wait level avoids the processing of the same quantities of good and bad programs previoulsy processed. The smaller rate of program arrival offsets the degradation of database services that had happened due to the greater amount of bad programs.

The condition "returned data buffer > 0" is observed by the absence of this value and by the absence of the value $very\ bad$ to the variable "buffer efficiency", since there is a correspondence between them. The other states obtained from this scenario show possible intermediate levels and different final equilibrium levels.

Buffer memory changing simulation

In order to run simulations using the memory agent shown in Figure 4, we create three sub-types of model fragments by assigning the agent values plus, minus and zero, in order to represent, respectivelly, increase, decrease and stable memory situation. The simulation shown in Figure 11 considers the three possibilities, under the condition that SQL efficiency process is balanced, that is, [good programs rate = bad programs rate].

State 1 shows a situation in which memory is decreasing. Few outcomes are possible due to the following constraining conditions: (i) [returned data buffer > 0] and (ii) [when IO device wait = too much then good programs and bad programs rate = 0]. Due to memory decrease, the derivative of *bad programs* and *good programs* is negative, showing a decrease of database requests due to the degradation of its services.

Another situation is the one starting in state 3 that shows a number of possibilities coming from the increase of memory available for buffer. One of these behaviour paths is shown in Figure 12.

As a consequence of memory increasing, it is possible to execute more programs, since the buffer efficiency has improved, as revealed by the positive sign of the derivative of variables good and bad programs. SQL efficieny remains unchanged since the decrease of IO device wait equally affects bad and good programs. Memory increase progressively improves buffer efficiency, as well as database response efficiency.

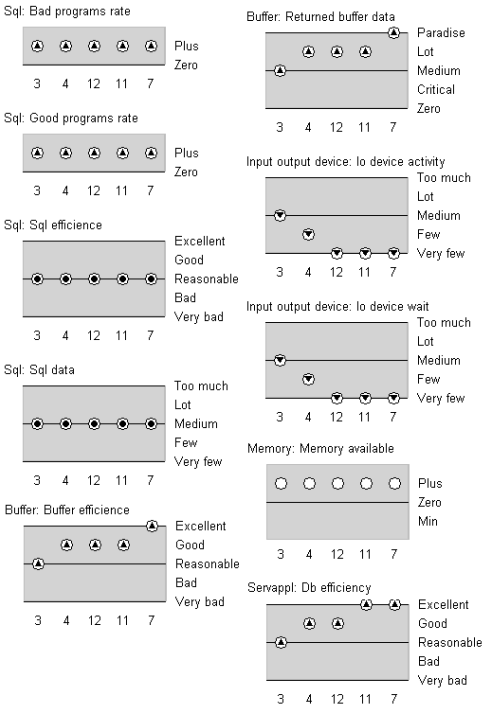

Figure 12: Value histories describing changes in quantities along the behaviours path initiated in state 3, in a simulation with buffer memory increasing.

IO device updating simulation

The third simulation shows the effects of an external agent acting for updating IO devices (disks). For a single data set to be read, the device activity is greater for obsolete disks than for new ones. As a consequence, obsolete disks cause greater IO wait than new disks, with respect to the same amount of IO requirements. As shown in Figure 5, we create a Device Upgrade Agent, which implements the IO devices updatings rate.

This process influences both the amount of up dated and the amount of non-up to date IO devices. An inverse correspondence was created beween the amount of upgraded IO devices and obsolete IO devices, because of their complementary values.

In order to obtain from the simulation all the possible behaviours engendered by the IO devices updating process, three sub-types of model fragments for the Upgrade Agent were created, reflecting the following situations: (a) the IO devices updating is carried out under a positive rate - more up dated IO devices increases; (b) there is a negative rate - obsolete IO devices increases and up dated ones decreases; and (c) the rate shows an equilibrium between updating and obsolescence of devices.

The simulation (Figure 13) produces all possible behavious considering of the three possible situations *IO device up-*

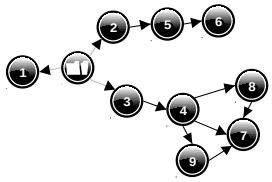

Figure 13: Behaviour graph of the IO device upgrade agent simulation.

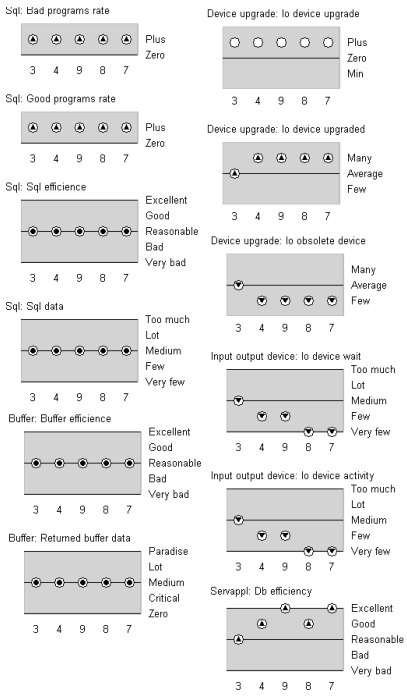

Figure 14: Value histories of behaviour paths initiated in state 3 - more device upgraded.

grade has value plus, zero and minus representing whether or not the IO devices updating process is being adequately accomplished, i.e., if the amount of devices becoming obsolete as years goes by is greater, equals or lower than the amount of up to date devices.

The set of value histories in Figure 14 shows possible outcomes of the situation described in state 3 ([IO device upgrade = plus]). This way, the simulation emphasizes what may happen if equipment is updated.

Three behaviour paths are possible [3 → 4 → 7], [3 → 4 → 9 → 7] and [3 → 4 → 8 → 7]. One might notice that the decrease of IO devices activity (for the same amount of requests) decreases the IO wait time and, as a consequence, allows more good programs nd bad programs to be executed in the environment.

Discussion

The main goal in constructing and simulating the model described in this paper is to illustrate the causal relation existing between internal parts of the Database Server and to highlight the various possibilities of equilibrium that might be reached without the need of interrupting systems' services or without the need of reaching an excellent level. Decrease or increase in the performance of an internal component part may propagate to other parts of the system by means of some causal relation. In some points of such causal chain, system performance may become critical and the whole system is likely to stop. These behaviour changes in the Database Server may affect the behaviour of related components, for example, the Application Server. However, as shown by the simulations, there is great deal of different equilibrium situations, in which different levels of service can be achieved before stopping the whole system.

According to the ontology adopted for this work, the QP theory, processes are the primary cause of changes in the system. Three processes were included in this model: (i) good and bad programs influencing SQL efficiency, (ii) management of buffer memory, and (iii), technological updating of IO devices. The effect of these processes may propagate and affect the database services behaviour and other components that depend on them. Improving the whole system requires adequate management.

Process (i) follows from the amount of requests addressed to the database by the Web application Server, specifically by bad and good programs aiming at responding to new business needs. The amount of requests follows from the usage level of the services offered by the financial bank, via Web pages. Therefore, a number of activities related to this process are under the bank's control (for example, the development of good and bad programs), while other activities are beyond bank's control, such as the amount of users (and programs fired by them) addressing requests to the database.

Process (ii) is usually under the supervision of a system operation support team that determines changes in the amount of memory available for buffer, according to the amount of memory remaining in the server that runs the database.

Process (iii) is usually under the responsibility of a hardware-storage team, which determines device updating according to availability of resources for investment and to machines life-time.

Lack of knowledge on how these three processes interact and may affect of the considered processes affect the database performance may lead to situations as those described by the model simulations. The worse case, not considered in the simulations, is the one in which simultaneous changes occur in all the three processes, while the distincted teams ignore what is happening within the environment. According to the change management discipline from ITIL (van Bon 2005), every change within the environment should be assessed by those in charge of a specific process with respect to its impact, better time for tasks to be accomplished, and other factors. However, difficulties are expected because change management rules recommend that impacts are assessed in accordance to the requiring area, and each area is only able to foresee impacts following directly from its actions, that is, within its expertise. Integration of different areas is urgently required.

The construction of the presented model and the interpre-

tation of simulations confirm our initial intuition and lead us to formulate the following hypothesis: awareness about causal chains relating components that shape the bank's computational system could support decision making of teams responsible for maintaining the environment, as well as help them to foresee incidents that may interupt computational services offered by the bank.

In order to improve the model, one possibility is to model the inner parts of the Application Server and the Web Server, providing an inner causal chain for each component, and also to model how possible behaviour changes the efficiency of the Database Server might affect the Application Server, or yet the Application Server affects the Web Server and vice-versa.

Concluding remarks

In this paper we argue on the potencial of qualitative modelling approach to deal with capacity and incident management of Information Technology services. We describe a problem firstly in terms of services offered by a financial bank having its Web Server, Application Server and Database Server as components that allow customers to request services via Web pages. A qualitative model was developed considering yet two related components, internally located in the bank's IT infra-structure, namely, Application Server and Database Server, along with three processes that influence the database performance. These are degaged from deploying the Database Server component into three internal parts. Out of these three processes, model simulations are presented and then interpreted.

Model construction and simulation analysis gave us initial elements to assess the potential of qualitative modelling to address IT services management, providing to different teams of workers a view of the services they have to regulate. This might aid support experts to disseminate technical information.

The model presented in the paper for illustrating the activities of a Database Server along with its causal relations may be refined. There are many other relevant components, and many other processes that might be supervised on the thread from data request to the Database Server up to its response. As the goal here is to open up a discussion on the application of QR to the field, we are aware that model extension and evaluation is still needed to confirm our initial assumptions. That is part of our ongoing work.

Instead of replacing existing techniques for capturing and treating messages from IT components, the use of QR for IT services management, as suggested in this paper, intends to complement the set of existing tools to improve management. The explored building blocks, as well as those that might yet be built, consider automation already existent in the environment, and also look foreward to envisage what could yet be (semi-)automated with the use of AI techniques for supporting human work, and for helping understanding of the environment's behaviour.

Finally for as far as we can see now, maintenance and control of technological services and the implementation of effective IT management are goals pursued by many companies nowadays (beyond banks), and are among their priorities. As such, further efforts might be invested for investigating under what conditions qualitative modelling of Web transactions and other services might be exploited by banks and other companies.

Acknowledgements

Ricardo Alcantara is grateful to Banco do Brasil for the support provided during his master science course.

References

Armstrong, E.; Ball, J.; Bodoff, S.; Carson, D. B.; Evans, I.; Green, D.; Haase, K.; and Jendrock, E. 2005. *The J2EE 1.4 Tutorial*. Sun Microsystems. http://java.sun.com/j2ee/1.4/docs/tutorial/doc/.

Bouwer, A., and Bredeweg, B. 2001. Visigarp: Graphical representation of qualitative simulation models. In Moore, J. D.; Redfields, G. L.; and Johnson, J. L., eds., *Artificial Intelligence in Education: AI-ED in the Wired and Wireless Future*. Amsterdam (The Netherlands): IOS Press.

Bredeweg, B., and Struss, P., eds. 2003. *Current Topics in Qualitative Reasoning*, volume 24.

Bredeweg, B.; Bouwer, A.; Jellema, J.; Bertels, D.; Linnebank, F.; and Liem, J. 2006. A new workbench for qualitative reasoning and modelling. In *ECAI Workshop MBS06 (submitted)*.

Bredeweg, B. 1992. *Expertise in qualitative prediction of behavior*. Ph.D. Dissertation, Department of Social Science Informatics, University of Amsterdam, Amsterdam (The Netherlands).

Falkenhainer, B., and Forbus, K. 1991. Compositional modeling: finding the right model for the job. *Artificial Intelligence* 51:95–143.

Forbus, K. 1984. Qualitative process theory. *Artificial Intelligence* 24:85–168.

Heller, U. 2001. *Process-oriented Consistency-based Diagnosis: Theory, Implementation and Applications*. Ph.D. Dissertation, Technical University of Munich.

Machado, V. B., and Bredeweg, B. 2002. Investigating the model building process with HOMER. In Bredeweg, B., ed., *Proceedings of the International Workshop on Model-based Systems and Qualitative Reasoning for Intelligent Tutoring Systems*, 1–13.

Salles, P., and Bredeweg, B. 2003. A case study of collaborative modelling: building qualitative models in ecology. In Hoppe, U.; Verdejo, F.; and Kay, J., eds., *Artificial Intelligence in Education: Shaping the Future of Learning through Intelligent Technologies*, 245–252. Osaka (Japan): IOS-Press/Ohmsha.

Sanders, R. E. 2003. Basic performance tuning. *DB2 Magazine* 8(3).

van Bon, J. 2005. *IT Service Management: An Introduction Based on ITIL*. The Netherlands: van Haren Publishing.

Weld, D., and de Kleer, J., eds. 1990. *Readings in Qualitative Reasoning about Physical Systems*. San Mateo, CA: Morgan Kaufmann.