

# A decision support tool to optimize scheduling of IT changes

Rodrigo Rebouças, Jacques Sauvé, Antônio Moura  
Department of Computing Systems  
Federal University of Campina Grande  
Campina Grande, Brazil  
{rodrigor, jacques, antao}@dsc.ufcg.edu.br

Claudio Bartolini<sup>1</sup>, David Trastour<sup>2</sup>  
<sup>1</sup>HP Laboratories Palo Alto, USA  
<sup>2</sup>HP Laboratories Bristol, UK  
{claudio.bartolini, david.trastour}@hp.com

**Abstract**— Change management is one of the most critical processes in IT management. Some of the reasons are the sheer number of changes and the difficulty of evaluating the impact of changes on the IT services being provided. Through carrying out a survey with IT managers and practitioners, we have found that, among the activities performed during change management, change scheduling (allocating changes to change windows) is the most problematic one. In this paper we solve the change scheduling problem by using a business-driven approach that evaluates the impact of a change schedule in terms of the financial loss imposed on the service provider. Toward this aim, we model the impact of SLA violations when the implementation of changes is done after their deadline. A change scheduling optimization problem is then formalized and its solution is applied to a typical scenario. The results show that optimizing the scheduling of changes can result in significant savings to an IT support organization.

**Keywords**- *change management, business-driven IT management, service level management, Information Technology Infrastructure Library (ITIL), business metrics, modeling, performance evaluation, business impact, decision and negotiation support tools*

## I. INTRODUCTION AND MOTIVATION

The IT information library (ITIL, [1]), and in particular the IT Service Management framework (ITSM, [2]) that derives from it, are gaining mindshare among IT practitioners and arguably represent by now the most widely accepted approaches to IT service management. Among the processes that ITSM specifies for IT service delivery and support, change management has a central importance. The mission of change management is to ensure that standardized methods and procedures are used for efficient and prompt handling of all changes, in order to minimize the impact of any related incidents upon service. To achieve this requires a careful and considered approach to assessing risk, potential impact of change, the resource requirements, and the process for approving changes.

ITIL and ITSM stop short of defining control objectives for IT processes, and are complemented in this aspect by the COBIT framework [21] (Control Objectives for Information and related Technologies). COBIT defines key goal indicators and maturity models for a number of IT processes including change management (referred to in COBIT as “manage change”). Key goal indicators are aimed at measuring the

outcome of the IT processes. However, these indicators tend to concern the quality of the change management process (percentage of urgent changes, number of changes that were rolled back ...) rather than the effectiveness of the IT processes measured through the impact of these on the business that IT supports. The argument we develop in this article is that as changes are planned, scheduled, executed and evaluated, the business impact of the changes need always be taken into consideration, along with the constraints on the IT resources (people, technology and processes) that need to be made available to carry out said changes. We aim at addressing this need through decision support tools for change managers.

State-of-the-art software tools for changes managers help them manage the change management process, by making sure that changes are authorized, by ensuring that all relevant parties have properly researched and documented all change procedures, and by coordinating the effort involved in building, testing and implementing changes.

From the results of a survey that we carried out in earlier 2006 with IT change managers and practitioners [3], the top 3 challenges in change management that current tools do not address clearly emerged as: 1) scheduling and planning of changes, 2) handling high number of urgent changes, and 3) dealing with ill-definition of request for changes. We have started a research program to address all these issues in order, starting with the scheduling and planning of changes.

In a previous work, [4] we have begun to formalize a mathematical method to approach business-driven planning and scheduling of changes. In [4], a formal method was developed to compare the business impact of individual changes as they disrupt interdependent services. We now use this result to solve the more complete change scheduling problem, by considering change dependencies, change windows during which changes may be performed, a set of changes to be implemented and the business impact of the overall schedule. In this work we will also describe a prototype software decision support tool that embodies the method and discuss the results of our optimization when applying the tool to a change management scenario.

## II. A DECISION SUPPORT TOOL FOR CHANGE MANAGEMENT

We intended to base our decision support tool on solid design principles. In this section we walk the path through the

This paper was published at 10th IFIP/IEEE International Symposium on Integrated Network Management (IM 2007), 21-25 May 2007 - Munich, Germany

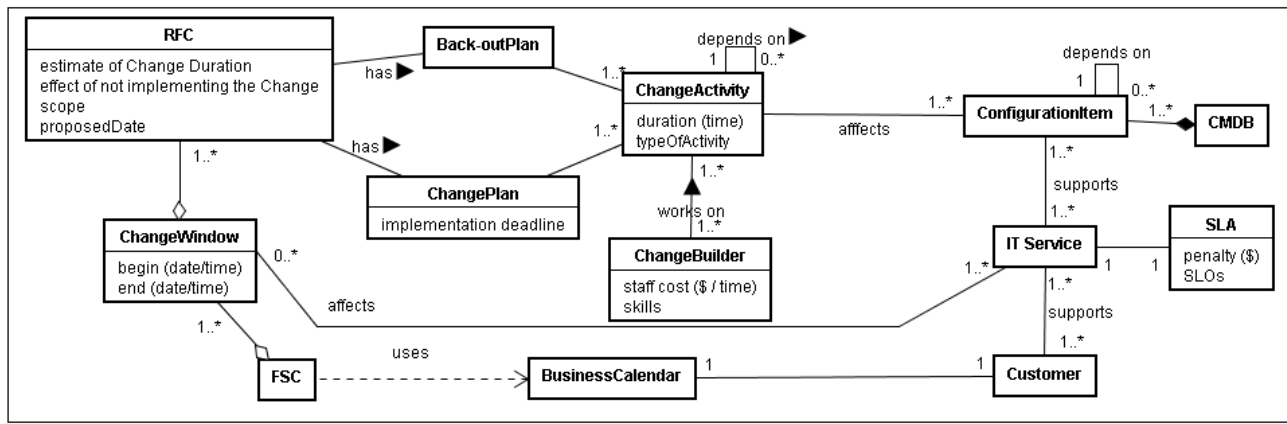


Figure 1. Change Management conceptual model

analysis and design phase of our software tool. The approach we follow is to carry out the analysis phases by looking at what change managers do in their daily job, and in particular what decisions must they make at any point during his supervision of the change management process. The result of the analysis phase is reported in this paper in the form of a conceptual model of all the relevant objects and attributes. Even though the analysis was worked out with the problem of optimizing the scheduling of changes in mind, the concepts that we have mapped have a wider applicability and we envisage that they become useful when we turn our attention to the next phases of our research program. Based on the conceptual model, we subsequently break down the problem of scheduling changes so as to minimize business impact – along the lines of what was done in [4] – and pose the basis for formalizing such a problem as a constrained optimization problem; this formalization is completed in the next section.

#### A. A day in the life of a change manager

The job of a change manager is to oversee the change management process throughout all its phases, from the initial classification that follows the submission of requests for change (RFC) to the review of implemented changes. The RFC is the only mechanism in ITIL for requesting changes to the infrastructure. A RFC must contain all the information necessary for a change to be assessed, approved and built.

As change initiators (usually technical staff) submit RFCs for consideration, the change manager filters and allocates initial priority to them. If the change is urgent, it will be dealt with through the urgent change management procedures. If the change is not urgent, then the next step is to check whether there is a change template (in ITIL parlance: *standard change model*) for this type of change. A change template represents a predefined way or procedure of dealing with simple, repeated changes of a known type or complexity. The aim of a change template is to facilitate the accurate and timely assessment of changes by the appropriate group of people. If such a change template exists, the change manager will follow the steps for initiating that procedure. Else, the change manager will have to indicate the type of change, and how to deal with it (impact,

cost, number of people needed to build it and the time it will take to build).

The change advisory board (CAB) is responsible for assessing the impact of requested changes and estimating the resource requirements. They will advise the change manager on whether changes should be approved and will assist in scheduling changes. CAB membership will depend on the change being requested and could consist of anyone who is potentially impacted by the change.

During a CAB meeting, a set of RFCs must be allocated to previously-negotiated change windows in order to minimize impact on the affected IT services.

#### B. A conceptual model for change management

From the previous subsection, we can start building a conceptual model for change management, which will be the basis for the information model underpinning our software tool. Where possible, we have tried to follow ITSM term definitions. The model is represented in Fig. 1.

The natural entry point to the model is the request for change (RFC). An RFC represents a formal proposal for a change to be made. It includes an estimate for the change duration, the foreseen effects of implementing and not implementing the change, and may contain information on the scope and the proposed date by which the change is requested to be completed.

An RFC is associated with a *change plan* and a *back-out plan*. A change plan is a document that identifies a series of *activities* and all the *resources* (people, technology, processes) that are required to implement the change. Similarly, a back out plan indicates activities taken to restore the situation as it was before undertaking the change, should something go wrong. The change plans also specifies an *implementation deadline*, by which the change must be implemented. Penalties may apply if not.

A change activity represents a set of actions that must be performed in order to complete a change implementation. The change activity might depend on other activities and are implemented by a *change builder*. The change activity may

affect one or more *configuration items*. The activity has an associated expected *duration*.

A change builder is a technician whom the change manager holds responsible to implement the change. The model associates a monetary hourly *cost* to each change builder.

A configuration item (CI) – following the ITSM definition – is any component that needs to be managed in order to deliver an IT service. Information about every CI is recorded in a record within the configuration management database (CMDB). Information recorded about the CIs typically include hardware, software, buildings, people and formal documentation such as process definitions or service level agreements (SLAs). CIs are subject to change control, and throughout their lifecycle all the relevant information about changes performed on them is kept up to date, care of the configuration management process.

A *change window* is an agreed-upon time when changes may be implemented, usually with minimal disruption on IT services. Change windows are documented in SLAs.

Finally, the *forward schedule of changes* (FSC or simply *schedule*) contains details of all the changes approved for implementation and their proposed implementation date. The FSC covers a set of change windows.

### C. Scheduling of Changes in Change Windows

In this section we introduce the problem that we set out to solve with our decision support tool. The problem is the optimal (according to some business-related criteria that will be specified later) scheduling of changes in change windows. To work out the constraints to this scheduling problem, we first observe that change activities are executed by the change builders, which represent a scarce resource and are specialized in some certain types of activities. Moreover their employment comes at a cost, which as we will see is bound to have an impact on the business criteria that are expressed in the objective functions of our optimization problem. The problem is further constrained by dependency relationships between the CIs that the change activities touch.

Figure 2. depicts the three steps into which we decompose the problem. The first step uses the information about the affected configuration items and the set of changes to be implemented and provides the dependency graph to the third step. The second step provides the change windows available to implement the changes. The third step allocates the changes to the change windows.

#### 1) Step one: Change dependency definition

Each change has a plan. A plan is composed of a set of implementation activities to be carried out by change staff members working on CIs. Functional dependencies may exist between activities; these dependencies can influence the sequence in which changes are implemented. Activity dependencies are influenced by CI dependencies, staff restriction, infrastructure availability, financial restrictions, etc. This work does not model the details involved in these dependencies but assumes that the resulting change dependencies are known. For example, if change  $c_i$  affects a database CI and change  $c_j$  ( $i \neq j$ )

calls for upgrading a server CI – which happens to support the database CI, then change  $c_j$  should be implemented first.

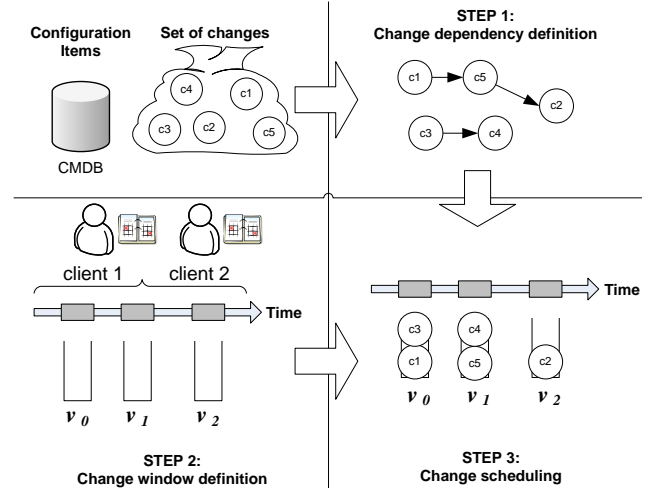


Figure 2. Steps for change scheduling

#### 2) Step two: Change window allocation

Based on the clients' business calendars, one must determine common time windows during which changes that affect one or more clients may be implemented. For instance, if change  $c_1$ , affects both clients A and B, whose business hours are (9 am to 5 pm) and (7 am to 11 pm) respectively, then a common time window during which  $c_1$  could be implemented could be, say, from 11 pm to 7 am the next day.

#### 3) Step three: Change scheduling

Given a set of changes, their dependencies and a set of change windows, this step allocates the implementation of each change to an available change window. The resulting schedule aims to minimize the business impact of implementing the set of changes.

This work focuses on step three: *change scheduling*. We assume that changes and their dependencies are given and that the change windows have previously been negotiated and defined. In the future, we plan to provide solutions to help the change manager perform step two.

### D. Formalization of the change scheduling problem

The change scheduling problem requires all changes to be allocated to pre-negotiated change windows, while minimizing some measure of impact to the business. Let the set of changes considered by the change manager be  $C = \{c_1, \dots, c_{|C|}\}$ . The change manager must allocate all changes in  $C$  to a given set of change windows  $W = \{w_1, \dots, w_{|W|}\}$ . Change window  $w_k \in W$  has a duration  $\Delta w_k$ , which is the total time interval available to perform changes within this change window.

Associated with each change  $c_n \in C$  are an expected duration  $\Delta c_n$  and a change plan, which defines a set of activities  $A_n = \{a_{n,1}, a_{n,2}, \dots, a_{n,|A_n|}\}$ . To each activity  $a_{n,i} \in A_n$  is associated its expected duration  $\Delta a_{n,i}$ .

Let us introduce the following notation to capture relevant timestamps in the lifecycle of a change  $c_n$ :

- $t_{n,0}^c$  is the time at which the RFC for change  $c_n$  appeared;
- $t_{n,I}^c$  is the time instant at which change  $c_n$  implementation begins;
- $t_{n,F}^c$  is the time at which implementation is concluded. Observe that  $t_{n,F}^c = t_{n,I}^c + \Delta c_n$ .
- $t_{n,D}^c$  is a deadline by which time the change must be implemented.

We now formalize the change scheduling optimization problem. All dependencies between activities are given as input and are represented by  $D^A = \{d(a_x, a_y), a_x, a_y \in A\}$ , where  $A = \bigcup_{n=1}^{|C|} A_n$  is the set of all activities, and  $d(a_x, a_y)$  means that  $a_y$  depends on (i.e. must be performed after)  $a_x$ . Activity dependencies lead to change dependencies  $D^C = \{d(c_x, c_y), c_x, c_y \in C\}$ , where  $c_y$  depends on  $c_x$  if and only if there is at least one activity  $a_i$  to implement  $c_y$ , that depends on at least one activity  $a_j$  to implement  $c_x, i \neq j$ .

Within a change window, changes are to be executed according to an *implementation order* (representing a partial order relationship between begin and end time of changes). For simplicity we assume the implementation of changes to be sequential, i.e., no changes are implemented concurrently. We define a *schedule*  $K$  be an allocation of changes to change windows, that is,  $K = \{k(1), k(2), \dots, k(n), \dots, k(|C|)\}$  where  $k(n)$  is the index of the change window to which change  $c_n$  was allocated, or  $c_n \in C_{k(n)}^w$ , where  $C_k^w \subseteq C$  is the set of changes that are allocated to change window  $w_k$ .

For a schedule  $K$  to be valid, it must take into account change dependencies and not exceed the total duration of any change window by assigning too many changes to it.

The goal is to assign each change in set  $C$  to a change window in  $W$ , so as to minimize an objective function which we denote as  $L(C, W, K)$  and which – without loss of generality – is made to represent a monetization of the loss to the business due to the impact of changes to the infrastructure. We thus have the optimization problem shown in Table I

TABLE I. CHANGE SCHEDULING PROBLEM FORMALIZATION

|                    |  |
|--------------------|--|
| <b>Minimize:</b>   | $L(C, W, K)$ , the total financial impact on the provider, for the set of changes $C$ and the set of change windows $W$ , when using schedule $K$  |
| <b>Over:</b>       | $K$ , the schedule   |
| <b>Subject to:</b> | The set of changes fit in each change window:<br>$\sum_{\forall n c_n \in C_k^w} \Delta c_n \leq \Delta w_k, w_k \in W$ Change dependencies must be obeyed:<br>$t_{x,F}^c \leq t_{y,I}^c, \forall d(c_x, c_y)$ |

### III. BUSINESS-RELATED CRITERIA TO DRIVE THE CHANGE MANAGER'S DECISIONS

In the previous section we gave the mathematical formalization of the problem that our decision support tool is solving. In this section, following the BDIM methodology introduced in [5], we derive a suitable representation for the business impact function that drives the optimization of the change schedule.

#### A. Estimating the business impact of changes

##### 1) Loss for a single change

The financial impact on the service provider associated with the implementation of a single change may be broken down into three parts, as depicted in Figure 3. :

1. **Loss before change implementation:** some changes may have an associated direct or indirect financial impact even before the change is implemented. For instance, a new service only starts to generate revenue to the service provider after implementing the change that launches it; in this case, service revenue is not accrued while the change is pending. The rate of loss (\$/time unit) associated with change  $c_n$  prior to implementation is represented as  $\varphi_{n,BC}^c$ . The loss associated with the change before it is implemented is then:

$$L_{n,BC}^c = \varphi_{n,BC}^c (t_{n,F}^c - t_{n,0}^c) \quad (1)$$

2. **Cost of change implementation:** change activities have a staff cost. Let activity  $a_{n,i}$  of a change  $c_n$  have a cost rate  $\varphi_{n,i}^a$ . As a result, the implementation cost for change  $c_n$  is given by:

$$L_{n,I}^c = \sum_{i=1}^{|A_n|} \varphi_{n,i}^a \cdot \Delta a_{n,i} \quad (2)$$

We assume that all activities occur according to plan, including procedure repetition, if required. Other costs such as equipment acquisition and training are not considered here, for simplicity.

3. **Impact of implementing the change after the change deadline:** The effect of not implementing a change by its deadline depends on the affected CIs, the change itself, the service, the associated SLA, etc. In the model being developed here, we focus on impact for those CIs that make the service become unavailable if changes that affect it have not been implemented by the deadline. One may argue that the deadline exists for a strong reason and that service may be considered to be “out of spec” after the deadline, hence unavailable. The impact of implementing a change  $c_n$  after its deadline passes is denoted by  $L_{n,D}^c$ . Section B will expand on this discussion and will provide an expression for this loss.

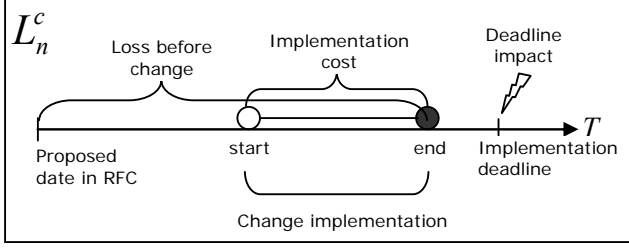


Figure 3. Loss for a single change

The loss,  $L_n^c$ , associated with change  $c_n$  is given by:

$$L_n^c = L_{n,BC}^c + L_{n,I}^c + L_{n,D}^c \quad (3)$$

### 2) Loss for a single change window

Change window  $w_k$  starts at time  $t_{k,I}^w$  and ends at time  $t_{k,F}^w = t_{k,I}^w + \Delta w_k$ , see Figure 4.

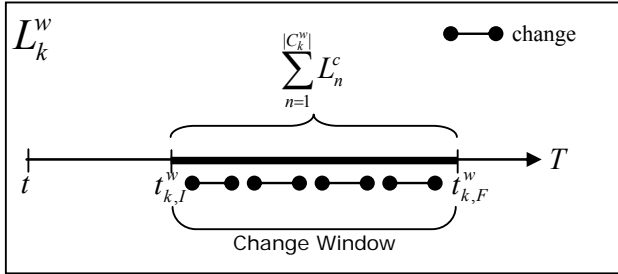


Figure 4. Loss composition for a single change window

The financial loss incurred by the provider for all changes in the change window is simply the sum of loss associated with all changes allocated to the window:

$$L_k^w = \sum_{\forall n|c_n \in C_k^w} L_n^c \quad (4)$$

### 3) Loss for several change windows

The total loss of the service provider, given a set of changes  $C$ , a set of scheduled change windows  $W$ , and a schedule  $K$ , is:

$$L(C, W, K) = \sum_{k=1}^{|W|} L_k^w = \sum_{n=1}^{|C|} L_n^c \quad (5)$$

### B. Estimating loss due to downtime

In order to solve the above change scheduling problem, one must now get an expression for  $L_{n,D}^c$  in equation (1).

As observed above, we assume that a service becomes unavailable if a change that affects it is not implemented by its deadline. Service unavailability may cause SLA violation. Let  $S = \{s_1, \dots, s_{|S|}\}$  be the set of services offered by the service provider. In order to understand what follows, let us focus on the current SLA evaluation period. Without loss of generality, let it last from time 0 to time  $T$ . The change manager must perform change scheduling several times over this period since RFCs may appear at any time. Let us focus on the situation at time  $t$ , the present, at which point the change manager has several changes to schedule over future change windows. The main point to be understood now is that the scheduling decision depends on what has happened in the past to service availability. If, in the time period  $[0, t]$ , a service has already accumulated some downtime, partially jeopardizing the SLA clause dealing with availability, then scheduling changes that will bring the service down may cause it to violate the SLA more easily. This must be taken into account by the change manager.

In a past paper [4], we have shown that if, at time  $t$ , service  $s_j$  has past mean availability over the time period  $[0, t]$  given by  $A_j(0, t)$ , then the probability of violating the SLA for service  $s_j$  is:

$$V_j(t, T, A_j^{\min}) = F_j \left( \frac{A_j^{\min} T - A_j(0, t)t}{T - t - \Delta T_j^s} \right) \quad (6)$$

where  $A_j^{\min}$  is a Service Level Objective (SLO) on availability of service  $s_j$ . In this equation,  $F_j(x)$  follows the two-parameter Beta distribution with parameters  $\alpha$  and  $\beta$ . The mean value for availability is  $\alpha/(\alpha+\beta)$ . We also borrow the business impact model from [4], which is applicable to providers receiving a fee for services and paying penalties for SLA violations. We therefore have the following expression for the potential loss for service  $s_j$  evaluated with the information available at the present time  $t$  and for the evaluation time  $T$ :

$$L_j^s(t, T) = V_j(t, T, A_j^{\min}) \cdot \pi_j + \gamma_j \cdot (\Delta T_j^s) \cdot \sigma_j \quad (7)$$

The loss is the sum of the potential penalty of service  $s_j$  plus the lack of revenue during the unavailability period  $\Delta T_j^s$ . The notation is explained in Table II.

TABLE II. PARAMETERS IN EQUATION (5)

|                         |   |
|-------------------------|---|
| $V_j(t, T, A_j^{\min})$ | Probability of SLA violation, given the knowledge available at time $t$                               |
| $\Delta T_j^s$          | Service $s_j$ unavailability period due to implementation of change after its implementation deadline |
| $\pi_j$                 | Penalty (\$) for service $s_j$ SLA violation  |
| $T$                     | SLA evaluation period for $s_j$   |
| $\gamma_j$              | Session throughput for service $s_j$ in transactions/unit time  |
| $\sigma_j$              | Fixed fee (\$) per successful transaction, for service $s_j$  |

Implementing change  $c_n$  can affect service  $s_j$  availability if it affects CIs that supports that service. Let service  $s_j$  be supported by a set  $I_j^s$  of configuration items. If we let  $I = \{i_1, \dots, i_{|I|}\}$  be the set of all CIs in the CMDB, then  $I_j^s \subseteq I$ . Each change plan specifies, among other things, a subset  $I_n^c \subseteq I$  of the CIs that will be affected by change  $c_n$ .

The service unavailability period  $\Delta T_j^s$  is composed of all intervals during which service  $s_j$  becomes unavailable within  $T$ , discounting all change window intervals within them (since the provider is not penalized for service disruption during change windows). See Figure 5.

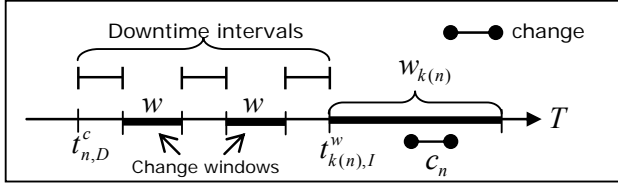


Figure 5. Downtime intervals caused by change  $c_n$  implementation after its deadline

Let  $i(x,y)$  be a time interval that starts at time  $x$  and ends at time  $y$  and let  $\Delta i = y - x$  be its duration. Let  $i_k^w = i(t_{k,I}^w, t_{k,F}^w)$  be the time interval corresponding to the duration of change window  $w_k$ . So, the time interval from change  $c_n$ 's deadline to the end of its implementation is given by  $T_j^s$  (see Figure 5. ):

$$T_j^s = \bigcup_{\forall n | I_j^s \cap I_n^c \neq \emptyset \text{ and } t_{k(n),I}^w > t_{n,D}^c} i(t_{n,D}^c, t_{k(n),I}^w) \quad (8)$$

To find the time intervals during which service  $j$  is marked as being down, one must discount change window intervals from  $T_j^s$ :

$$T_j^s = T_j^s - \bigcup_{\forall k | i_k^w \subseteq \Delta T_j^s} i_k^w \quad (9)$$

The service unavailability period  $\Delta T_j^s$  is given by the duration of the set of intervals  $T_j^s$ .

At time  $t$  (when changes are being scheduled), the value of  $L^s(t, T)$ , the total expected provider financial loss due to downtime for all services during the current SLA evaluation period ending at time  $T$  is therefore, given by:

$$L^s(t, T) = \sum_{n=1}^{|C|} L_{n,D}^c = \sum_{j=1}^{|S|} L_j^s(t, T) \quad (10)$$

From equations (1) and (3), we have:

$$L(C, W, K) = \sum_{n=1}^{|C|} L_n^c = \sum_{n=1}^{|C|} (L_{n,BC}^c + L_{n,I}^c) + \sum_{n=1}^{|C|} L_{n,D}^c \quad (11)$$

or, the final result for  $L(C, W, K)$ :

$$L(C, W, K) = \sum_{n=1}^{|C|} (L_{n,BC}^c + L_{n,I}^c) + L^s(t, T) \quad (12)$$

This completes the theoretical development. In the next section we apply the method to an e-commerce scenario and show the different business impacts from running the model through a prototype.

#### IV. APPLICATION OF THE CHANGE SCHEDULING TOOL TO AN E-COMMERCE SCENARIO

Our scenario assumes the IT infrastructure depicted in Figure 6. Before any changes are made, the infrastructure supports a credit card payment service (AS-IS IT infrastructure). Plans are made to upgrade the infrastructure by deploying an additional e-commerce service (TO-BE). To bring the infrastructure from the AS-IS to the TO-BE status, a number of requests for changes are made (the twelve changes are listed in table III).

Each service is subject to an SLA. The configuration items that support each service are grouped as follows: the credit card service is supported by a billing server and an application server, which are behind firewall B; the e-commerce service will be supported by a new e-commerce server, a new database server and the existing router and firewall A.

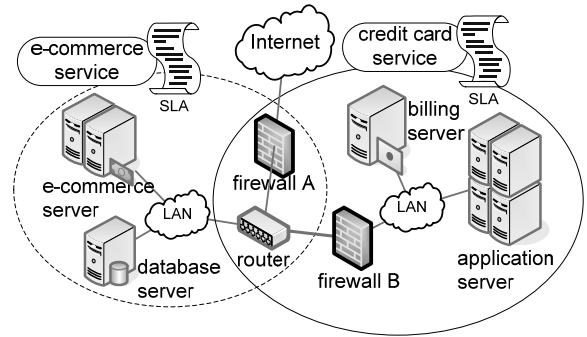


Figure 6. An e-commerce scenario

##### A. The changes

Let time be counted in days from day one. The current infrastructure that supports the credit card service must be updated by day 28 and the new e-commerce service must be available by day 20 (the *service start date* stated in its availability SLA). To achieve these business objectives, the changes in table III must be implemented. The changes include hardware installation, security patches and software upgrades.

Changes 1 and 2 are software upgrades for router and firewall A. Their implementation does not affect any service availability. Changes 3 and 4 are software upgrades to the servers supporting the credit card service; their deadlines are set at day 30. Change 5 will install a new firewall for the credit card infrastructure and must be implemented by day 28. Changes 6 and 7 will install the new servers that will support the e-commerce service. They must be implemented by day 20. Change 7 has a particularity: from the date it was planned (day 9) to its implementation, the provider has a daily cost of \$ 300, due to an Internet link that has already been upgraded. Changes 8 to 12 are security patches that must be applied to servers and router. If they are not implemented by their deadlines, service will come down for security reasons.

Change dependencies may be derived from implementation activity dependencies, affected CIs, assigned personnel, and so on. Estimating change duration is carried out from the durations of implementation activities. For simplicity, all activities are omitted from the example.

TABLE III. CHANGES TO THE EXAMPLE INFRASTRUCTURE

| #  | Description      | Deadline | Dep. | Dur. | Affected CI(s) | Affected service(s)    |
|----|------------------|----------|------|------|----------------|------------------------|
| 1  | Software upgrade | 20       | -    | 4h   | Router         | -                      |
| 2  | Software upgrade | 30       | -    | 4h   | Firewall A     | -                      |
| 3  | Software upgrade | 30       | -    | 4h   | App. server    | credit card            |
| 4  | Software upgrade | 30       | -    | 4h   | Bill. server   | credit card            |
| 5  | Install          | 28       | -    | 4h   | Firewall B     | credit card            |
| 6  | Install          | 20       | -    | 8h   | DB server      | ecommerce              |
| 7  | Install          | 20       | -    | 8h   | E-comm. server | ecommerce              |
| 8  | Security patch   | 20       | 6    | 2h   | DB server      | ecommerce              |
| 9  | Security patch   | 20       | 7    | 2h   | E-comm. Server | ecommerce              |
| 10 | Security patch   | 28       | 3    | 2h   | App. server    | credit card            |
| 11 | Security patch   | 28       | 4    | 2h   | Bill. server   | credit card            |
| 12 | Security patch   | 20       | 1    | 2h   | Router         | ecommerce, credit card |

Now, let's position ourselves at day 10. The change manager must schedule the 12 changes in four previously-negotiated change windows, shown in Table IV along with their start day and duration.

TABLE IV. CHANGE WINDOWS

| Change Window | Start day and time        | Duration |
|---------------|---------------------------|----------|
| 1             | Tue. 14 at 5 PM           | 15h      |
| 2             | Sat. 18 at 9 PM           | 15h      |
| 3             | Mon. 21 at 5 PM           | 15h      |
| 4             | Fri. 25 at 8 AM (holiday) | 15h      |

The e-commerce and credit card services are bound by an SLA which defines a 99% availability SLO. At scheduling time (day 10) the accumulated availability (obtained from

management software) of the credit card service is 99.2%. Since the e-commerce service is still being implemented, its past availability is set to 100%. The penalty for availability SLO violation is \$ 15,000.00 for the e-commerce and \$ 40,000.00 for the credit card service. Measured service throughput for both services is 5 sessions per second. Each successful session generates a fee of \$ 1.00 from the e-commerce and \$ 0.50 from the credit card service for the provider. Both SLAs are evaluated at day 30. See table V.

TABLE V. SERVICE PARAMETERS

| Parameter         | E-commerce     | Credit card    |
|-------------------|----------------|----------------|
| $A^{\min}$        | 0.990          | 0.990          |
| $A^{\text{past}}$ | 1.000          | 0.992          |
| $\pi_j$           | \$ 15,000.00   | \$ 40,000.00   |
| $T$               | 30             | 30             |
| $\gamma_j$        | 5 sessions/sec | 5 sessions/sec |
| $\sigma_j$        | \$ 1.00        | \$ 0.50        |

The change manager needs to schedule the set of changes (obtained from the CMDB) to the set of available change windows (negotiated with the affected customers) in order to reduce the provider's loss.

### B. Optimization Results

Running our tool against the simple example in the previous subsection we note that the provider's potential loss arising from the scheduling of all 12 planned changes varies widely: from a few thousand to a few million dollars! Table VI offers samples of possible change schedules out of all possibilities of scheduling the 12 changes to the available 4 windows. The brackets represent the change windows and the numbers inside them are the changes allocated to each window.

TABLE VI. SAMPLES OF OPTIMIZATION RESULTS

| Sample # | Scheduling                      | Potential loss (\$) |
|----------|---------------------------------|---------------------|
| 1        | [1,7,9][6,8,12][2,3,4,10][5,11] | \$ 7,896.00         |
| 2        | [1,6][7,8,9,12][4,11][2,3,5,10] | \$ 7,896.00         |
|          | (...)                           |                     |
| 3        | [1,6,8][7,9,12][2,3,4,10][5,11] | \$ 9,096.00         |
| 4        | [1,7,9][2,3,4,12][5,6,8][10,11] | \$ 453,913.62       |
| 5        | [1,7,12][2,3,4,10][5,6,8][9,11] | \$ 1,911,913.62     |
|          | (...)                           |                     |
| 6        | [6][4,5,11][1,2,3,10][7,8,9,12] | \$ 2,897,300.00     |

Window 1 in sample #2 uses 12 hours of 15 available hours. It could accommodate change 8 (security patch for the database server), but the dependency between changes eliminates it. The database server security patch cannot be implemented before the database server is installed.

Sample #3 shows the impact of the loss caused by the available but unused Internet link. Its potential loss (\$ 9,096.00) is a result of implementing change 7 on the 18<sup>th</sup> (instead of the 14<sup>th</sup>), during change window 2.

The potential losses from samples #4, #5 and #6 result from implementing changes after their deadlines: changes 6 and 8 in sample #4; changes 6, 8 and 9 in sample #5; and changes 7, 8, 9 and 12 in sample #6.

Full results (not shown here) indicate that 75 schedules, out of a total of 100,020 viable schedules, offer minimal potential loss. There are 3,644 worst case schedules (with potential loss of \$ 2,897,300.00). Choosing a high loss schedule through ad hoc means is very likely and thus risky. The proposed optimization approach mitigates such risk. When there are few changes to be scheduled to few windows, the change manager could examine all minimal schedules provided by the algorithm and select those considered cheaper (not all implementation costs have been considered in this example) or a schedule that is more comfortable for the technical staff. Comfort could be estimated in terms of fewer work hours to implement changes in a given window (over a holiday or a weekend, say). Notice that sample #1 causes the team to work only 6 hours over the weekend (versus 14 hours in sample #2)! This could be a real added bonus, given the stressful conditions change management teams operate under.

### C. Computational complexity of the solution

To solve the example, we used a brute-force search algorithm that examines all schedules in the schedule space. Since each change can be allocated to one of  $|W|$  windows and since there are  $|C|$  changes, there are  $|W|^{|C|}$  schedules to consider. Therefore, the computational complexity of the algorithm is  $O(C \cdot |W|^{|C|})$ . For scenarios consisting of up to about 15 changes, the time to find a solution is of the order of a minute or so on a common PC. For larger problem sets, heuristics are needed and we leave this for the future.

### D. About the prototype

The developed prototype is structured as shown in Figure 7. It is composed of four components (*Scheduler*; *Schedule Generator*; *Schedule Constraints*; *Service Downtime Calculator* and *Loss Calculator*) and one artifact (*Scenario Description*).

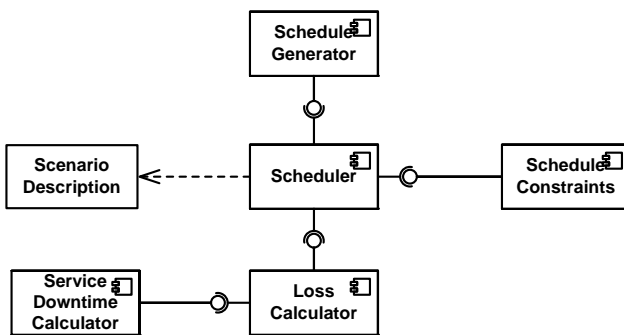


Figure 7. Prototype architecture

The *Scenario Description* is a text file that describes:

- the changes to be implemented: each change has a change plan with the change description, creation time,

duration, implementation deadline, change dependencies and affected services;

- the change windows available; each has an initial time and duration;
- the services supported by the affected configuration items: for each service information about technical characteristics is described (throughput, availability at the present time); SLA (penalty, evaluation time and agreed-upon availability) and the fee paid for each successful transaction.

The *Scheduler* receives the *Scenario Description* as input and orchestrates execution using the other three components which it depends on.

The *Schedule Generator* generates all possible schedules of the described changes on change windows. Each generated schedule is filtered by the *Schedule Constraints* component, which defines the constraints that define a valid schedule. Potential loss is calculated for all valid schedules by the *Loss Calculator*. The *Service Downtime Calculator* calculates the effective service downtime of all affected services of a valid schedule; this information is used by the *Loss Calculator*.

## V. APPLICABILITY AND LIMITATIONS OF THE CURRENT PROTOTYPE – FUTURE WORK

The development of our prototype for change scheduling is ongoing, and it is part of a more comprehensive and longer-term research program on decision support for IT change management. In this paper we extend the mathematical model that we began sketching in [4] and we build a software embodiment of it, which we apply to an e-commerce scenario. However, the state of the development is such that some simplifying assumptions remain which we will address progressively going forward with our program.

To begin with, we have made the simplifying assumptions that changes occur sequentially and that the assignment of change activity to change builder is known upfront. However, change activities are often executed concurrently in large IT organizations. Also, activities are not directly assigned to change builders; in practice, there is a level of indirection and activities are assigned to workgroups (Unix group, database group, etc.). It is clear that concurrent changes are important and will need to be dealt with in a future version of the tool. Nevertheless, we believe that the insights gathered with the current version of the tool are useful and will be instrumental in producing more elaborate models in the future. As an example of insights, we can show how widely different schedules can affect the business.

It also has to be noted that the direct applicability of the current prototype is restricted to normal changes, for which a change plan exists. It does not cover urgent changes, which must be treated in quite a different way, since services may be down as a result of infrastructure problems and it may not be appropriate to wait for a change window to execute the change.

Regarding the algorithm for scheduling, from the observations on its computational complexity made in section IV-C we conclude that we need to branch out another thread of

work addressing the development of heuristic algorithms. Real-life scenarios derived from interviews with enterprises in the banking sector consider a number of changes that can easily be in the hundreds per weekly change window – possibly over a thousand over a time horizon of a month, and we envisage that the optimization algorithm used in the example will not scale up to what is required for such scenarios.

## VI. RELATED WORK

Our work relates to those in software tools in general; to those applying Business-driven IT Management concepts to IT Service Management; and to IT Change Management in particular.

The state of the art of commercially available software is such that a number of IT management software tools are present in the market today that provide administrative support to change managers. However, although these tools do provide valuable support throughout the change lifecycle, none of them address the problem of driving the decision of the IT managers via business objectives.

Only recently has IT Service Management attracted the attention of the network and system management research community. Regarding change management in particular, Keller et al. [6] were the first – to our knowledge – to address this space with CHAMPS. The CHAMPS system provides automation for the change manager, whereas in this work we address decision support aspects and support for negotiation of the forward schedule of change in CAB (change advisory board) meetings, thereby providing a nice complementarity between the work presented here and CHAMPS.

Some recent works have addressed the effect of business-related considerations in providing automation and decision support for IT Service management. For a comprehensive review of the state of the art in business-driven IT management, see [5]. The more relevant to our line of research among them concern processes that are contiguous to change management such as incident management, problem management, service level management and the other processes related to IT service support. The process of service level management lends itself quite naturally to bringing business-related considerations to IT management. Notably, [7],[8],[9],[10],[11] among other works, address the problem of minimizing business impact of service level violation. Later on, [12] have applied a very similar methodology to the one described in this paper to prioritization of incidents, deriving utility functions from business objectives and developing a comprehensive framework and a method for incident prioritization that takes into account strategic business objectives such as total customer experience thereby going a long way towards the much needed alignment of IT and business objectives.

Works on driving IT service *delivery* processes (as opposed to service *support*) from business considerations are somehow more established in the academic literature. Among others we could cite [13][14][15] for workload management and resource allocation. Similarly, in autonomic computing systems, “Utility functions are an attractive candidate for a much-needed lingua franca for high-level objectives” [16], and works such as [17]

have beaten this path. However, all of these approaches are directed at the use of business objectives (or, more generally, utility functions) to achieve closed-loop control of managed system. Our work differs substantially from these approaches in that the decision problems we solve are aimed at providing decision support to humans, and the conclusions drawn from our software represent suggestions that they (change managers in this case) may at any time supersede.

Finally, Jeng et. al [18] can be cited as a representative approach using logic-based policies rather in the tradition of [19][20] for what they call business performance management, which is fundamentally different from our quantitative approach.

## VII. CONCLUSION

In this paper we presented a methodology and a software prototype for decision support to optimize scheduling of IT changes driven by business considerations. Change management is a process of fundamental importance for IT service support, and from a survey that we carried out earlier this year (2006) with IT managers and practitioners, it comes out clearly that planning and scheduling of changes is a pain point that is not addressed by any of the commercially available tools.

We gave an account of the analysis and design phases in the inception of our change scheduling prototype, and presented a deep dive into the formalization of the decision problem that our tool solves: how to define a schedule for allocating changes to pre-negotiated change windows while minimizing the expected negative impact of service disruptions on the business that the IT department supports.

We applied our prototype to an e-commerce scenario, and presented optimization results that speak volumes about the value that such tool could bring to the business. We concluded by discussing the applicability and the limitations of the current prototype, and how we intend to address the limitations as future work.

## ACKNOWLEDGMENT

The authors would like to thank Abdel Boulmakoul, Maher Rahmouni and Katia Saikoski for their valuable input and comments.

This work was developed in collaboration with HP Brazil R&D.

## REFERENCES

- [1] IT Infrastructure Library, “ITIL Service Delivery” and “ITIL Service Support”, Office of Government Commerce, UK, 2003.
- [2] Van Bon, J., Chief Editor, “IT Service Management, an introduction based on ITIL”, itSMF Library, Van Haren Publishing, 2004.
- [3] The HP-Bottom Line Project, “IT Change Management Challenges – Results of 2006 Web Survey” Technical Report DSC005-06, Computing Systems Department, Federal University of Campina Grande, Brazil, March 2006. [http://www.bottomlineproject.com/bl\\_media/techreport/005-2006.pdf](http://www.bottomlineproject.com/bl_media/techreport/005-2006.pdf)
- [4] Sauv e J, Rebou as R, Moura A, Bartolini C, Boulmakoul A, Trastour D, “Business-driven support for change management: planning and scheduling of changes”, In: 17th IFIP/IEEE International Workshop on

- Distributed Systems: Operations and Management (DSOM 2006), October 23-25, Dublin, Ireland, in press.
- [5] Sauv e, J., Moura, A., Sampaio, M., Jornada, J. and Radziuk, E., "An Introductory Overview and Survey of Business-Driven IT Management", in Proceedings of the 1<sup>st</sup> IEEE / IFIP International Workshop On Business-Driven IT Management, in conjunction with NOMS 2006, Vancouver, Canada, pp. 1-10.
- [6] Keller, A., Hellerstein, J., Wolf, J.L., Wu, K. and Krishnan, V., "The CHAMPS System: Change Management with Planning and Scheduling", In *Proceedings of the IEEE/IFIP Network Operations and Management Symposium (NOMS 2004)*, IEEE Press, April 2004, pp. 395-408.
- [7] Liu, Z., Squillante, M.S. and Wolf, J.L., "On maximizing service level agreement profits" In proc. ACM Conference of Electronic Commerce, 2001
- [8] M. J. Buco, R. N. Chang, L. Z. Luan, C. Ward, J. L. Wolf, and P. S. Yu, "Utility computing: SLA management based upon business objectives", IBM Systems Journal, 2004.
- [9] M. Sall e, C. Bartolini, "Management by Contract", In Proc. Network Operations and Management Symposium (NOMS 2004), IEEE Publishing, 2004.
- [10] Sauv e, J., Marques, F., Moura, A., Sampaio, M., Jornada, J. and Radziuk, E., S :A , "Design from a business perspective", In Proc. 16th International Workshop on Distributed Systems: Operations and Management (DSOM 2005).
- [11] Issam Aib, Mathias Sall e, Claudio Bartolini, Abdel Boulmakoul, Raouf Boutaba and Guy Pujolle, et al. "Business Aware Policy Based Management", IEEE/IFIP 1st Workshop on Business Driven IT Management (BDIM) 2006, April 2006, Vancouver, Canada.
- [12] Bartolini, C. and Sall e, M., "Business-driven prioritization of service incidents", In proc. 15th International Workshop on Distributed Systems: Operation and Management (DSOM 2004).
- [13] Sauv e, J., Marques, F.T., Moura, A., Sampaio, M., Jornada, J., Radziuk, E., "Business-Oriented Capacity Planning of IT infrastructure to Handle Load Surges", In Proc. IEEE/IFIP Network Operation and Management Symposium (NOMS 2006).
- [14] Bruno Abrahao, Virgilio Almeida, Jussara Almeida, Alex Zhang, Dirk Beyer, Fereydoon Safai, "Self-Adaptive SLA-Driven Capacity Management for Internet Services", IEEE/IFIP NOMS Conference, Vancouver, Canada, April 2006
- [15] X. Zhu, Z. Wang, S. Singhal, "Utility Driven Workload Management using Nested Control Design", in American Control Conference, ACC 2006
- [16] J.O. Kephart, "Research Challenges in Autonomic Computing", Proceedings of the 27th international conference on Software engineering, 2005
- [17] W. E. Walsh, G. Tesauro, J. O. Kephart, and R. Das. "Utility functions in autonomic systems". In Proceedings of the First International Conference on Autonomic Computing, pages 70–77. IEEE Computer Society, 2004.
- [18] Jun-Jang Jeng, Henry Chang, Kumar Bhaskaran: Policy Driven Business Performance Management. DSOM 2004
- [19] Nicodemos Damianou, Naranker Dulay, Emil Lupu, Morris Sloman: The Ponder Policy Specification Language. POLICY 2001: 18-38
- [20] Naftaly Minsky, "Law Governed Interaction (LGI): A Distributed Coordination and Control Mechanism (An Introduction, and a Reference Manual)", Rutgers University Technical Report
- [21] IT Governance Institute, "Cobit 4th Edition", 2006, [www.isaca.org/cobit.htm](http://www.isaca.org/cobit.htm)