Management by Contract

Mathias Sallé, Claudio Bartolini
Trusted Systems Laboratory
HP Laboratories Palo Alto
HPL-2003-186(R.1)
May 28, 2004*

E-mail: {firstname.lastname@hp.com}

Management by Contract, contract, SLA, IT management, service level management

Companies are increasingly seeking to manage their IT infrastructure in light of their business objectives. To address this need, we introduce Management by Contract (MbC) as a new paradigm for IT Management. We propose a way of formalizing and analyzing contractual relationships in order to better inform IT-related decisions. In this paper, we present the information model that underpins MbC and describe our contract-based analysis algorithm.

Management by Contract

Mathias Sallé, Claudio Bartolini
HP Research Laboratories
1501 Page Mill Road
Palo Alto, CA 94304
{firstname.lastname@hp.com}

Abstract
Companies are increasingly seeking to manage their IT infrastructure in light of their business objectives. To address this need, we introduce Management by Contract (MbC) as a new paradigm for IT Management. We propose a way of formalizing and analyzing contractual relationships in order to better inform IT-related decisions. In this paper, we present the information model that underpins MbC and describe our contract-based analysis algorithm.

Keywords
Management by Contract, Contract, SLA, IT Management, Service Level Management

1 Introduction

After periods of significant growth, IT spending is now flat, oscillating around 3.3% of the revenue of an enterprise [12]. In this tight environment, businesses increasingly look for bottom line value delivered from IT. In that setting and in response to large investments driven by IT initiatives, companies have expressed a clear need to manage flexibly and efficiently their existing infrastructure and internal operations in light of their business environment. Their objective is to assess IT related issues in light of the business value that they entail and use this information in their decision making process.

IT management is generally achieved through four major steps: observe the IT events occurring in the enterprise, interpret them, take decisions and act accordingly. In order to better inform the decision making step, current approaches, such as Business Activity Monitoring (BAM), provide real-time access to critical business performance indicators. Although necessary, these approaches are often focused on internal business and IT metrics, not taking into consideration the business context within which an enterprise interacts.

We introduce *Management by Contract (MbC)* as a new paradigm to look at IT decisions in the context of their business environment. We suggest that this paradigm is necessary to transform the IT function to deliver superior business value at an acceptable cost level.

Our approach is based on the analysis that the implication of taking actions is to bear their associated consequences, be they positive or negative. This entails that no rational decision-making can be made without a clear understanding of the consequences of a decision.

Management by contract starts with the observation that service provision and receipt is often governed by an agreement where the parameters of the service, and the associated penalties and rewards for both parties are defined. To that, it adds the intuition that knowledge about one enterprise's objectives is captured and can be extracted from contracts and SLAs that were negotiated by the enterprise.

We propose a way of formalizing and analyzing contractual relationships to understand the impact that a decision would entail. We adopt a utilitarian and deliberative approach to objectives thus allowing a decision-maker to look at meeting or not meeting them as alternative viable business options.

We illustrate MbC through a scenario presented in section 2. We then give in section 3 an overview of the system and its information model. In section 4, we present our algorithm for contract-based analysis. We give experimental results in section 5 and describe related work in section 6. Lastly, we conclude this paper in section 7.

2 Scenario

In order to set the stage for the rest of the paper, we describe a scenario that illustrates the use of *Management by Contract* to drive IT-related decisions based on their business value.

We assume a dynamic e-business infrastructure for an on-line car part retailer CarParts.com. We focus on the internal business process that supports CarParts.com delivery service.

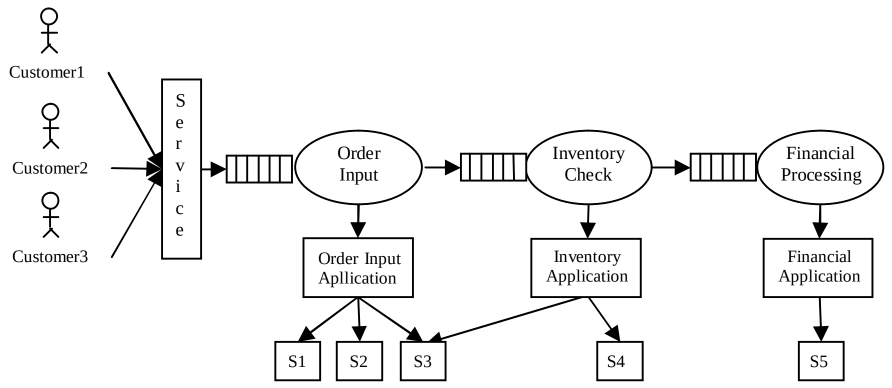


Figure 1: CarParts.com business process and its mapping onto the underlying IT infrastructure

2.1 Business process

The example business process is divided into the three stages of: *order input*, *inventory check* and *financial transaction processing.* Figure 1 depicts CarParts.com's business process and its mapping onto the underlying IT infrastructure.

The work at each node is carried out by an application that runs the business logic required to perform the necessary steps. In turn, the applications are running on enterprise servers. As an example, the inventory check node depends on the homonymous application, which in turn depends on the servers S3 and S4. Notice that some servers can run more than one application, as it is the case for S3 in Figure 1, which supports both the order input and the inventory check.

2.2 Customers and SLAs

CarParts.com's customers are segmented into three categories: Gold, Silver and Bronze, to which different qualities of service are guaranteed. The SLAs provided in our example govern the time between order and shipment, and are defined as follows:

- **Gold SLA**: Time between order and shipment shall be less than 3 days; otherwise the cost of the order is fully refunded.
- **Silver SLA**: Time between order and shipment shall be less than 5 days; otherwise a refund of 10 % of the cost of the order or $70, whichever is greater will be applied.
- **Bronze SLA**: Time between order and shipment shall be less than 10 days; otherwise $50 will be refunded.

2.3 Roles

Three roles are involved in this scenario. We focus mainly on the Business Operations Manager who is responsible for the management of business processes. Supporting the Business Operations Manager, the Systems Manager is responsible for managing the IT infrastructure and associated services. Lastly, and less involved in this scenario, the Strategic Manager is responsible for setting the overall strategic direction and priorities of the business.

Our scenario focuses on the effect of an IT failure (server S4) on the operational flow, and support for the Business Operations Manager to make appropriate operational decisions to handle the situation.

2.4 Failure and management options

When server S4 goes down, the IT monitoring systems alert the Systems Manager and report the impact of the failure, namely that Inventory Check application is no longer operational. Along with this information the likely root cause for the failure is presented as analyzed by the root cause analysis system. Using this data and the management system knowledge base, the System Manager is able to plan alternative repair strategies, together with expected costs, durations and impact of these strategies on service availability. In this case, we imagine that he identifies 3 alternative options:

- **Option1 - Repair today**: external consultant charging $1000 per half-day work. This will result in the Inventory application being fully down for 4 hours.
- **Option2 - Repair tomorrow**: internal IT employee: internal charge of 500$. This will result in the application being fully down for 24 hours.
- **Option3 - Repair tomorrow, and temporarily migrate the application onto server S5.** At the same cost than Option2, this will result in the Inventory and Financial application each running at 50% capacity for 24 hours.

The System Manager also identifies which Business Operations Manager is potentially affected by the service degradation and reports the different repair options to them. The Business Operation Manager must then determine the effect of the different service degradation options on her current and expected workload, and make two decisions:

1. Which of the repair options she will recommend to the systems manager
2. How best to re-plan her workload to make the best use of the current reduced capacity

These are the kind of questions that an MBC system will help addressing. We will refer to this scenario in the remainder of the paper to exemplify the technical proposition. For now it will be enough to say that the space of possible decision is a complex one and many variables are to be taken into account. For example, just looking at the cost of executing the management option or minimizing the system downtime may **not** result in the preferred option when considering the business context as a whole.

3 Overview of the system

In this section, we present how *Management by Contract* is introduced in the current, state-of-the-art IT management stack.

3.1 Extending current IT Management Stack

IT Management can be abstracted into a three layer stack. At the bottom lies the *monitoring layer* which probes the liveliness of the IT systems and particular system or service parameters. When a system is down, a failure alarm is generated. Consequently, a list of impacted services is determined using meta-information such as service dependency hierarchy and service topology. When comparing service parameter measurements to given thresholds extracted from SLAs and in cases of non compliance, violation and degradation alarms are also reported.
Both types of alarm – fault and violation/degradation – are either reported directly to the decision maker (IT Manager, System Operator, etc.) or are used as input to the second layer of the stack, namely the *diagnosis* layer. The purpose of this layer is to identify causes of faulty behavior. Causes are then either

reported directly to the decision maker or possibly used as input to the third layer, the *recovery planning* layer which seeks to determine recovery plans for the input causes. As a result of this analysis, multiple options describing recovery plans and associated cost are determined. In current management systems either the options are presented to the end user, or the recovery planning layer tries to find the optimal recovery plans given an objective function defined over parameters reflecting IT concerns.

As depicted in Figure 2, our approach extends the current three layer model by adding a fourth layer, the *Contract-based Analysis Layer*. Its purpose is to give a business context to the options generated by the recovery planning layer. Based on the analysis on the business engagements and objectives, captured within the SLAs that the enterprise committed to, the contract-based analysis layer associated a utility value to each of the options. This utility reflects the overall impact that a recovery option would have on the use of the services impacted. It is obtained by analyzing the consequences of violating or complying with the various service level objectives agreed during the negotiation phase of the contract, and the costs associated with each option.

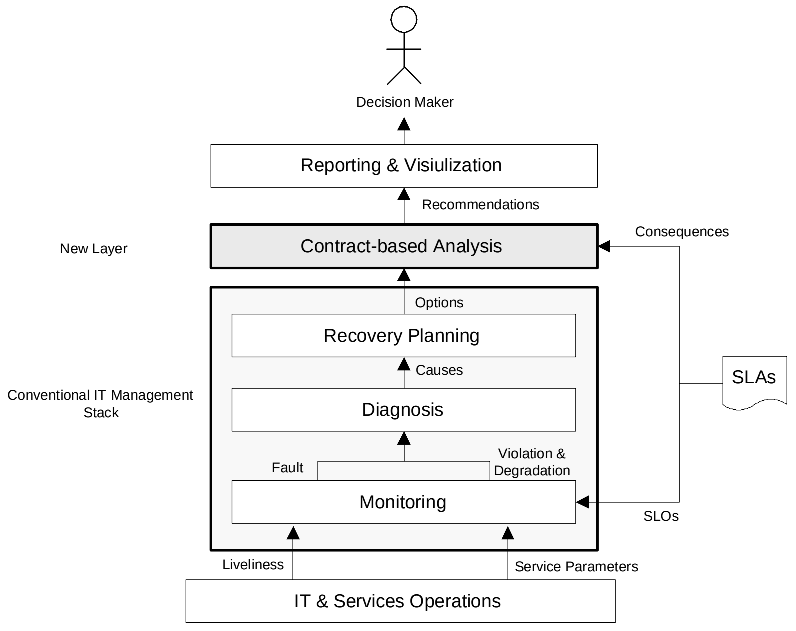


Figure 2: Extending the current management stack with MBC

3.2 Information Model

In the scenario in section 2, we applied *Management by Contract* to better inform repair decisions in the context of an on-line retail store that models its services as workflows. However, the scope of MBC is not limited to that setting and applies to various activities in different vertical domains where IT decisions have to be made in light of business information, e.g. resource allocation in Utility Data Center, bandwidth allocation in network provisioning etc. It is therefore a requirement for our underpinning information model to be as generic and as flexible as possible in order to be used in these various domains.

3.2.1 Service and Resource Model

We define our service and resource model to be generic by using abstract concepts. We furthermore show how it can be extended in the case of a simple workflow-based service. Other services would extend this core model.

We model resources in our IT infrastructure as *concrete resources*. Examples of concrete resources can be a Linux server, an instance of a RDBMS, etc. that are modeled through their own parameters. We then introduce the concept of abstract resources, as a mean to decouple the actual resource from the view that its clients (defined as instances of *ResourceClient*) must have of it.

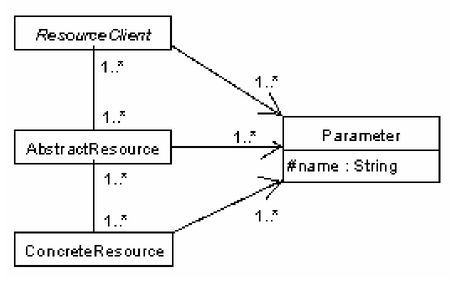


Figure 3: UML Diagram representing the Object Classes of the Resource Model

A ResourceClient models any kind of apparatus that possesses descriptive parameters and that uses IT resources; it can be a system, a process, an application, a service or business process, etc. A resource client is bound to one of more abstract resources, in the same way as an abstract resource can be bound to one or more concrete resources. The binding is obtained through the correct mapping of the parameters at every level.

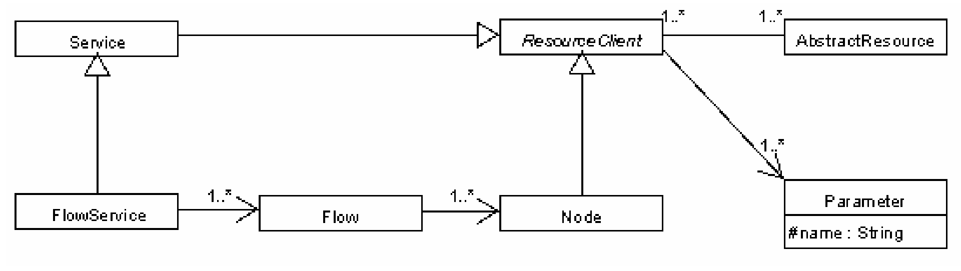


Figure 4: UML Diagram representing the Object Classes of the Service Model

A *service* is characterized by its service parameters. As in the scenario presented in section 2, some type of services use business processes. We refer to them as *flow services*. In our model, flow services are expressed through an ordered set of *nodes*. Types of nodes, their relationships, and languages to express them are beyond the scope of this paper. Nodes and services are defined as a kind of resource client.

Resource clients such as services and nodes have *scheduling policies* associated to them. Scheduling policies relate to the workload of the resource client. They represent a way of scheduling the workload by using a comparator over the (service) requests that are present in the workload at any given time. A workload is implemented as a queue system. When the scheduling policy is applied to a workload it has the effect of reordering the workload queue based on the comparator that it specifies. Requests refer to a service level agreement for any given customer.

3.2.2 Contracts and Service Level Agreements

As discussed earlier, SLAs and (more generally) contracts are used in MBC to provide business context to a decision making process. These SLAs can be defined over a large variety of services such as on-line video service, network provisioning, on-line retail shop, any type of web service, etc. However, their intent is always similar: to provide warranties over some parameters of a given service, penalties for not meeting these warranties and possibly rewards for exceeding them. Not only does this imply that the Contract/SLA model should be generic and flexible in terms of its descriptive capability, but it should furthermore be able to capture the dependency relationships (positive and negative consequences) that exist between clauses within a contract. Also, in order to derive utility from the analysis of these contracts, the model must adequately capture penalties and rewards.

The UML class diagram in Figure 5 describes the basis for the contract information model. A contract consists of a collection of clauses. Each clause states the *undertaking* that is promised, and the consequences of meeting and not meeting it. Consequences in turn take the form of clauses.

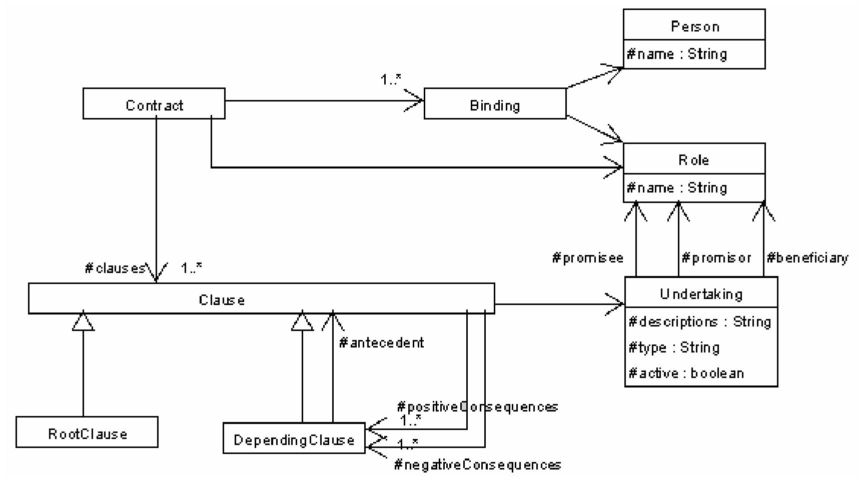


Figure 5: UML Diagram representing the Object Classes of the Contract Model

A contract also contains a collection of bindings between roles in the contract and the actual persons that play them. Examples of roles are buyer, service provider, etc.

Because of the dynamic nature of the business interactions that contracts model, not all the undertakings that are specified in a contract are *active* at a given time. Some of them will become active as time progresses, while some other may never become active, as in the case of penalties that never materialize. When contractual analysis is carried out, only the active undertakings and their consequences are taken into account.

As presented in the UML class diagram in Figure 6, undertakings are characterized by specific roles: promisor, promisee and beneficiary. The promisor is the role manifesting the intention; the promisee is the role to whom the promise is addressed; the beneficiary is the role other than the promisee that benefits from the performance of the intention.

Undertakings come in two kinds: promises of bringing about a certain state of affairs (a “seinsollen”, or “ought-to-be” undertaking) and promises of carrying out a certain contractual action (a “tunsollen”, or “ought-to-do” undertaking). A seinsollen undertaking specifies the state that is promised to be brought about through a predicate. A tunsollen undertaking specifies the action that is promised to be carried out.

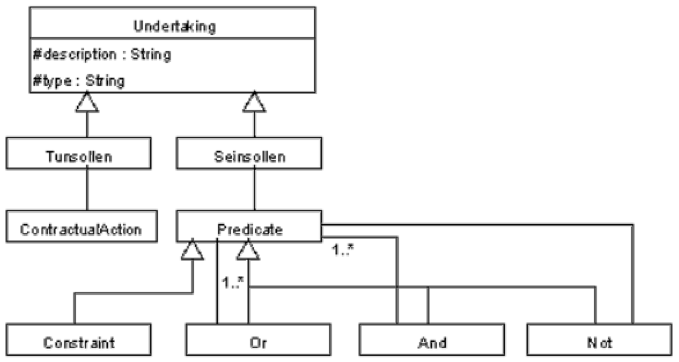


Figure 6: UML Diagram representing the Object Classes of the Undertaking Model

3.3 Service Level Agreements and Service Level Objectives

As depicted in Figure 7, a Service Level Agreement (SLA) is defined as a kind of contract. Our system requires it to specify the customer that the SLA refers to from the point of view of the MBC user. This information is instantiated by looking through the binding of the SLA and extracting the person representing the MBC user's counterpart in the SLA seen as a contract. An SLA is defined over a service.

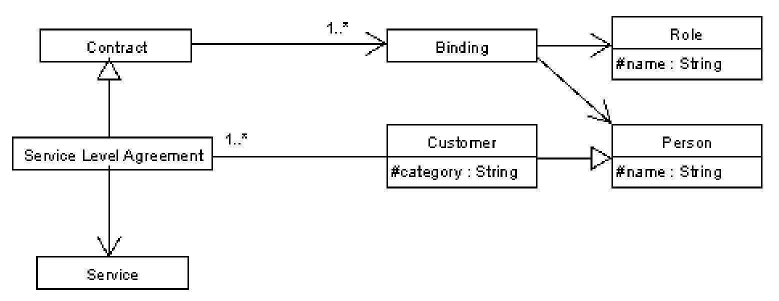


Figure 7: UML Diagram representing the Object Classes of the SLA Model

Service Level Objectives are modeled as seinsollen undertakings containing a predicate of type ServiceConstraints defined over ResourceClient parameters.

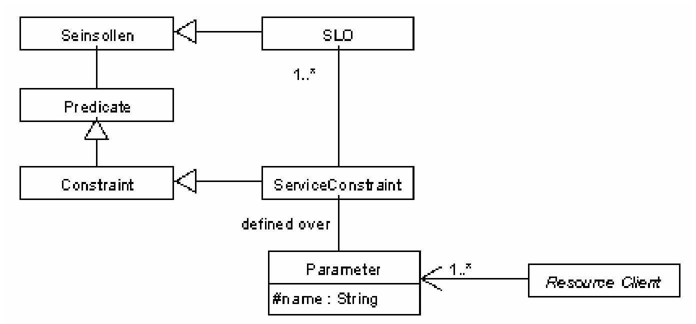


Figure 8: UML Diagram representing the Object Classes of the SLO Model

4 Contract-based Analysis

4.1 Contract Utility

Given a set of Service Level Agreements and a set of options, determining which option has the least impact on the business relationships is crucial to achieve management of IT resources in light of an enterprise business context.

Computing the contract utility of a contract is tantamount to assessing the value that an enterprise would perceive from a contract given an outcome, and that outcome's probability of occurring. Such an outcome could, for example, be the violation of a Service Level Objective. If the likelihood of violation is low, the perceived utility will be higher that if the likelihood is high. Similarly if the associated penalty is a function of the violation, the outcome itself will influence the perceived utility.

From a utility point of view, a contract or an SLA can be viewed as network of clauses: a clause has positive and negative consequences that are themselves clauses. The contract utility of an undertaking v given its likelihood of violation λ, is given by:

$$u_C(v, l) = (1 - l)(u_V + u_+) + l u_-$$

Here u_v is the direct utility of the undertaking v, u_+ is the utility of the positive consequences and u_- is the utility of the negative consequences.

In our previous scenario, if we consider the gold SLA: "**Time between order and shipment shall be less than 3 days; otherwise the cost of the order is fully refunded**" and if we consider an order worth $1000 of profit and an associated likelihood of violation of 0.2, the resulting contract utility would be:

$$(1 - 0.2) * 1000 + 0.2 * (-1000) = 600$$

4.2 Strategic Utility

Estimating the utility of an option focusing solely on the contractual utility might lead to incomplete results. We group under strategic utility the valuation of the expected outcome from enacting the option that falls outside the notion previously defined of contract utility.
We define three types of strategic utility:

- The *SLO strategic utility*
- The *customer strategic utility*
- The *enterprise strategic utility*

The SLO strategic utility captures the value of an outcome with regard to an objective defined as an SLO. For instance, all else being equal, an enterprise may tend to prefer to comply to an SLO with a strategic partner rather than with a second-tier partner. In this case, contractual information alone is not sufficient to evaluate the utility for a certain outcome; extra-contractual information needs to be taken into account, such as the perceived strategic value of each partnership, and the damage that either would suffer because of the contractual breach.
The customer strategic utility focuses on the value of a particular outcome independently of the SLOs that are active at a given moment. Take for example, an enterprise which has declared that its strategic objective is to always guarantee a certain degree of service availability to one of their preferred partners, regardless of what SLOs are in place with them. Therefore, extra-contractual considerations need to be factored in when computing the total utility of a management option that would result in a reduced level of availability for the customer.
The enterprise strategic utility focuses on the objectives defined by the enterprise independently of its contractual relationships. For instance, an enterprise might commit to the strategic objective of delivering on time for 95% or more of the orders.

The differentiation among the different kinds of utility plays a part in the algorithm presented in section 4.4 In general, strategic utility takes the form of a scalar function defined over the parameters that determine a certain outcome. More specifically, for the SLO strategic utility, the utility function definition might strongly depend on the violation likelihood for the SLO.

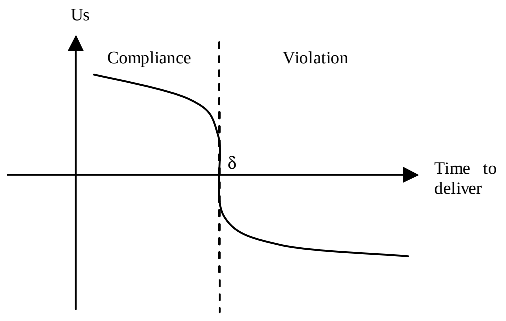


Figure 9: Example of strategic utility

For example, consider the shape of the SLO strategic utility function sketched in Figure 9. Two neatly separated regions can be identified in the figure: the *compliance* region, to the left of the deadline time, and the *violation* region to its right. The SLO strategic utility is slightly decreasing in each region, so as to signify that an earlier delivery is preferred to a late delivery. Across the deadline time the utility falls dramatically, as should be expected given the impact that meeting or violating the SLO might have on the satisfaction level of the customer.

4.3 Computing expected service level and likelihood of violation

To perform contract-based analysis, we need to determine the outcome of the various options to assess. For each impacted service, we first need to determine its expected *service level*. This service level is characterized by the expected value of the relevant parameters of the service.

SLOs are defined over service parameters. In turn, service parameters are expressed as functions of internal parameters (*internal mapping*). For instance, given a business flow, the service parameter *Time to Delivery* can be defined as the aggregate of the processing time for each node of the business flow. Internal parameters are themselves characterized by the availability of the underlying IT resources. For instance, the expected time of execution at a generic processing node might be characterized in terms of the availability of IT resources. Furthermore, each option entails a forecasted value of resource availability. We refer to this forecast as *Resource Availability Profile* (RAP). If, for instance, the option in our scenario in section 2 is to repair tomorrow, a RAP might be that for the three hours following the IT failure, the availability of the resources is 0. After that period, the availability increases to 50% and finally after 8 hours, the availability goes back to 100%.

Given an option and a scheduling policy, we compute the associated service level for an impacted service by determining the availability of resources using the option RAP and the relevant service parameter values using the internal mapping functions. Once the service level is determined, we compute the associated likelihood of violation.

SLOs mainly take the form of threshold constraints over the values of service parameters. Given a forecasted service parameter value pfv~$N(\mu_f,\sigma_f)$ and a threshold tv~$N(\mu_t,\sigma_t)$, we seek to compute the likelihood of violation, that is to say the probability that pfv > tv or pfv < tv, depending on the constraint operator.
Under the assumption that pfv and tv are independent, this is equivalent to computing the probability that a variable with normal distribution $(m_f - m_t, \sqrt{s_f^2 + s_t^2})$ is less (greater) than 0. This results in λ, the likelihood of violation of the SLO.

In order to calculate the likelihood of violation for SLOs that represent more complex expressions over service parameters, the same principles can be applied, resulting in correspondingly complicated calculations.

4.4 Algorithm

We now present a pseudo-code algorithm that shows how the various steps of the contract-based analysis are executed.

We assume a set O of options, a set S of scheduling policies, a set RC of impacted resource clients and suppose a workload Wrc associated to each impacted resource client rc in RC.

We use the following notation:

- ***u(o,s,rc,r,slo,*** l_{slo}***)*** denotes the utility of a request r associated to a resource client rc computed for a particular SLO and its likelihood of violation λ_{slo} given a specific option o and a scheduling policy s.
- ***u(o,s,rc,r)*** denotes the contract utility of a request r associated to a resource client rc given a specific option o and a scheduling policy s.
- ***u(o,s,rc)*** denotes the contract utility associated with a resource client rc given a specific option o and a scheduling policy s.
- ***u(o,s)*** denotes the contract utility associated with a specific option o and a scheduling policy s.

- ***u_c(r,slo, l_{slo})*** denotes the contract utility associated with a particular request r given an SLO and its likelihood of violation.
- ***u_s(r,slo, l_{slo})*** denotes the SLO strategic utility.
- ***u_s(r)*** denotes the customer utility.
- ***u_s(rc)*** denotes the enterprise utility.
- ***u_{imp}(o,s)*** denotes the utility associated with the cost of implementation of the option o and a scheduling policy s.
- ***U*** is the utility set containing all the utilities u(o,s) computed so far.

```
Begin
 Determine the set iSLO of impacted SLO
 For each Option o in O
   For each Scheduling Policy s in S
    For each ResourceClient rc in RC impacted
      Compute new workload NWrc by applying s to initial workload Wrc
      For each request r in NWrc
          For each slo in iSLO
             Compute expected service level $sl_{slo}$
             Compute likelihood of violation $\lambda_{slo}(sl_{slo})$
             Compute utility $u(o, s, rc, r, slo, l_{slo}) = u_c(r, slo, l_{slo}) + u_s(r, slo, l_{slo})$
          End
          Compute $u(o, s, rc, r) = \sum_{slo \in iSLO} u(o, s, rc, r, slo, l_{slo}) + u_s(r)$
      End
      Compute $u(o, s, rc) = \sum_{r \in NWrc} u(o, s, rc, r) + u_s(o, s, rc)$
    End
    Compute $u(o, s) = \sum_{rc \in RC} u(o, s, rc) + u_{imp}(o, s)$

    Add $u(o,s)$ to the utility set U
   End
 Return U
 End
End
```

Algorithm 1: Determining the utility of options based on contractual and strategic analysisAlgorithm 1 is a set U of utilities u(o,s). To maximize the utility of the decision making, the appropriate recommendation is adopt the option with the highest utility.

5 Experiment Results

We have carried out a series of experiments with the intention of evaluating the gain in utility due to the application of MBC in a simplified business scenario. To set up our experiments, we have used the scenario given in Section 2. With our experiments we investigate how much gain can be made just by applying the utility calculation to a few scheduling policies. For the sake of simplicity, we consider only one management option; therefore the resource availability profiles at each of the processing nodes do not change.

For a given option, we consider 4 different scheduling policies: First Come First Served (FCFS), Tightest Deadline First (TDLF), Highest Order Value First (HOVF), and Maximize Expected Utility (MEU). The MEU policy uses information about deadlines, gain and penalties relative to incoming requests, together with an estimate of the likely duration of request processing given the assumed resource availability profile at each of the nodes that compose CarParts.com's business process. MEU attempts, at each point in time, to maximize the overall expected utility for each of the possible orderings of the request queues at each process node.

Each of the runs of our experiments was conducted by simulating the arrival of over a thousands requests into the system. After the arrival of the 1000th request, we would take a snapshot of the three queues at each of the nodes (*order input*, *inventory* and *financial*), and the first 30 requests in the system would be considered. By fine tuning the parameters of arrival rate, and expected completion time at each node, the 30 resulting requests were distributed among the three process nodes. In particular, for the results that we present, mean time between requests arrival was set at 5 time unit, with an average amount of work for each request is picked at random from a uniform distribution centered on 8 work units, ranging between 4 and 12. In order to simplify the calculations involved, we assumed a resource availability profile at each node that would guarantee that one work unit would be executed in one unit of time. That meant that the average time for each node to process a request was also distributed uniformly between 4 and 12 time units. The experiments assumed that the value of the orders followed a lognormal distribution, with median 1000 currency units and positioned the likely maximum order[1] at 10000 currency units. For each order, the SLAs that we considered dictated a penalty of 10%, 50% or 100% (depending on the experiment) of the order value for late delivery. The utility derived from delivering on time was modeled by imposing a profit margin of 15% on the value of the order. The deadline for each request was also picked at random from a uniform distribution.

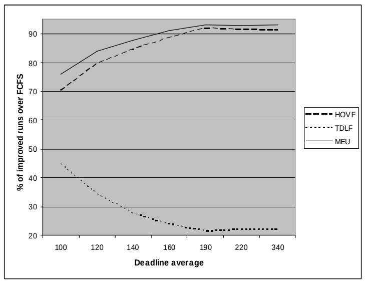
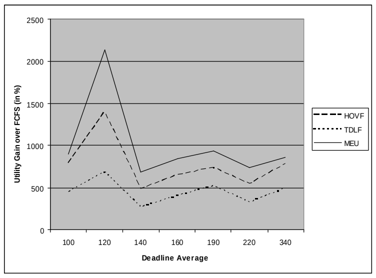
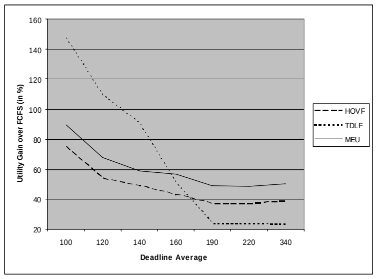


Figure 10: Percentage of improved runs, 10% penalty

Figure 10 shows the percentage of runs in which applying HOVF, TDLF and MEU resulted in a utility gain over FCFS when the penalty value is set to 10%. Similar trends are obtained when the penalty value increases.
We observe that increasing the deadline average leads to greater gains for the number of runs improved increases as the deadline increase when applying MEU and HOVL. This can be explained observing that raising the deadline average has the effect of leaving more leeway for rearranging requests, therefore resulting in a higher number of runs in which utility can be improved. However, when applying TDLF, the number of gaining runs *decreases* as deadline average increases. This is due to the fact that the TDLF policy only considers the deadline value and does not take into account the business context in which the enterprise operates. Finally MEU results on average in a 10% gain over HOVL.





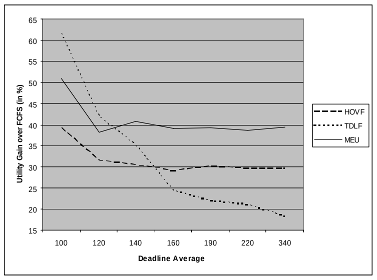


Figure 11: Utility Gain with 10% penalty

Figure 12: Utility Gain with 50% penalty

Figure 13: Utility Gain with 100% penalty

Figure 11, Figure 12 and Figure 13 show the variations of the utility gain, in relative terms, relative to FCFS for t HOVF, TDLF and MEU, as the deadline average increases and the penalty value changes.

[1] Likely maximum order being defined as a value with a 99% likelihood that an order extracted at random from the distribution would have a lower value than it.

We observe that although increasing the deadline results in a higher numbers of runs improved (Figure 10), the utility gain obtain by applying the three policies decreases. This is because as increasing the deadline average increases the number of requests expected to meet their deadline, resulting in an increasing utility gain for FCFS. As we compare the gains of the other policies to FCFS, as the deadline increases, less improvement in utility gain can be achieved.

It is interesting to note that when the penalty increases, the utility gains also decrease, since violating a deadline results in higher losses. Also, TDLF results in higher utility gains when the deadline average is small and the penalty rate is high, because this policy tries to minimize the violation at all cost. When the deadline increases, TDLF brings little utility gains over FCFS.

6 Related Work

Electronic Contracts and SLAs have been studied in various fields [1,10] ranging from e-commerce to network management. Various models have been proposed that focus on different stages of the e-contract lifecycle such as provision[3], execution[4,7,14], monitoring[6,8,9,13,14,16] and enforcement[6,8,9,14]. However, in general, these studies tend to only consider the part of the SLA concerning guarantees. As a result, the common behavior observed is of compliance with the prescriptions specified in the contract, which often leads to over-provisioning, deadlock in execution, and sanctions being imposed when in real life they might have not. Our approach takes an amoral and utilitarian approach to norms. Not complying with an obligation is considered as one viable business option. Our method carries out the analysis of the positive and negative consequences of complying, or not, with a contractual clause, and derives from it a suggestion as to how the enterprise's utility can be maximized.

Policy is often associated to contracts and SLAs. The DMTF [17] defines a policy-based framework for management where policies are expressed with a formalism derived from event-condition-actions rules. Sloman et al. [10] make an attempt to increase the flexibility, in the policies definition. However, and especially in the case of obligation type policies, their agents are purely reactive and are denied the opportunity to critically ponder the pros and the cons of complying or not with the norms. On the occurrence of an event, if the relevant conditions are evaluated true, the action associated to the obligation is automatically triggered by the platform. In contrast with these approaches, we propose to use contractual obligations as concepts to reason over rather than artifacts to specify the behavior of a management agent in a declarative way. This enables our model to be adaptive in that it responds more flexibly to a changing business environment by considering the consequences of norm adoption from an economic perspective.

In a different field of computer science, Decision Theory attempts to provide a rational basis for choice under uncertainty. Most frameworks rely on knowledge about an agent's preferences over an abstract set of possible outcomes, and assume the existence of probability and utility functions such that acting to maximize expected utility is rational in the sense that it respects those preferences. One of the biggest challenges that Decision Theory faces is the elicitations of the agents' preferences [15]. Our approach builds on this framework by adding to it the intuition that a rational enterprise's objectives and preferences are captured into contracts and SLAs, and knowledge about them can be extracted by applying contractual analysis in the way that we described in this work.

Finally, similar principles have been investigated. In Agent Theory, Boella *et al.* have proposed [2] a conceptual framework built on the Belief-Desire-Intention model that gives an agent the possibility to examine the relationship between intentions and obligations. They also include reasoning about the application of sanctions. In Resource Management, Menascé *et al.* have shown [11] how resource management in ecommerce websites can be driven by considering customers, their profiles and the amount of money accumulated in their shopping cart. Although they do not use contractual information they achieve management of IT resource in light of business context.

7 Conclusion

Based on the analysis that the implication of taking actions is to bear their associated consequences and starting from the observation that service provision and receipt is often governed by an agreement, we have

introduced in this paper a new paradigm, Management by Contract, to drive IT management from a business prospective.
We have proposed a way of formalizing and analyzing contractual relationships to understand the impact on the business that IT-related decision would entail. Experiment results show that adopting a utilitarian and deliberative approach to objectives results in considerable improvement in utility gain, both in relative terms and in number of improved cases.
Further work will include modeling different types of penalties, be they fixed penalties, penalties expressed as function of the violation time or right granted and analyzing their impact on the utility improvement that Management by Contract achieves.

8 Acknowledgements

We are grateful to A. Christodoulou and A. Byde for helpful discussions and valuable comments.

References

[1] S. Angelov and P. Grefen. B2B eContract Handling – A Survey of Projects, Papers and Standards. *University of Twente Library, CTIT Reports.*
[2] G. Boella and L. Lesmo. Deliberate normative agents. In R. Conte and C. Dellarocas, editors, *Social Order in Multi Agent Systems*. Springer Verlag, Berlin, 2001
[3] K. Czajkowski et al. SNAP: A Protocol for Negotiating Service Level Agreements and Coordinating Resource Management in Distributed Systems. *Proceedings of 8th Workshop on Job Scheduling Strategies for Parallel Processing, July 2002.*
[4] A. Daskalopulu and T. Maibaum. Towards Electronic Contract Performance. *Proceedings of International Workshop on Legal Information Systems and Applications Workshop, Dexa 2001.*
[5] N. Damianou, N. Dulay, E. Lupu, M Sloman. Ponder: A Language for Specifying Security and Management Policies for Distributed Systems. *Imperial College Research Report DoC 2001, Oct. 2000*
[6] A. Goodchild et al. Business Contracts for B2B. *Proceedings of ISD 2000.*
[7] P. Grefen et al. CrossFlow: Cross-Organizational Workflow. Management in Dynamic Virtual Enterprises. *IEEE Data Engineering Bulletin, 24, 2001.*
[8] M. Greunz *et al.* Supporting Market Transaction through XML Contracting Containers. *Proceedings. of the Americas Conference on Information Systems (AMCIS) 2000.*
[9] A. Keller et al. Managing Dynamic Services: A Contract Based Approach to a Conceptual Architecture. *Proceedings of NOMS 2002.*
[10] O. Marjanovic and Z. Milosevic. Towards Formal Modeling of e-Contracts. *Proceedings of 5th IEEE International Enterprise Distributed Object Computing Conference 2001.*
[11]D. Menascé and V.Almeida. Business-oriented Resource Management Policies for E-commerce Servers. *Performance Evaluation, September 2000.*
[12] Meta Group Worldwide Research Database, *July 2002.*
[13] A. Sahai *et al.* Automated SLA Monitoring for Web Services. *Proceedings of IEEE/IFIP DSOM 2002*
[14] M. Sallé and A.Boulmakoul. Integrated Contract Management. *Proceedings of HPOVUA Workshop 2002.*
[15] A. Schreiber *et al.* Knowledge Engineering and Management: The CommonKADS Methodology. *MIT Press(2000)* .
[16] L. Xu and M. Jeusfeld. A Concept for Monitoring of Electronic Contracts. *Infolab Technical Report Series 2003.*
[17] DMTF. www.dmtf.org