Management by Contract:
IT Management driven by Business Objectives

Claudio Bartolini, Abdel Boulmakou, Athena Christodoulou,
Andrew Farrell, Mathias Sallé and David Trastour

HP Research Labs
{firstname.lastname}@hp.com

Introduction

In today's business environment, change must be seen not as an exception but as the normal state of affairs. With its Darwin Architecture [23], HP addresses *business agility*, defined as the ability of one enterprise to respond to and anticipate business changes with speed and flexibility while meeting cost, quality of service and risk management objectives. One of the requirements that agility imposes on an enterprise is that their business groups and IT (Information Technology) groups must come together to create an *adaptive enterprise* in which business needs drive IT response, and business and IT changes are synchronized in real time. When projected onto the space of IT management, this requirement is translated into the need of managing IT in light of the business objectives of an enterprise.

We introduce *Management by Contract* (MbC) as a paradigm to address this need. In this work we describe the architecture of the MbC solution for IT management driven by business objectives.

The rest of this paper is structured as follows. We start by presenting the basic ideas of Management by Contract and we give a motivating scenario describing a hypothetical company that hosts IT infrastructure. Later we describe the architecture of our prototype system. We then move onto reporting on some experimental results and sketch the future directions of our research. We move onto the conclusions after a section that compares our work with related works in the literature.

The MbC Solution for IT Management driven by Business Objectives

MbC formalizes and analyzes contractual relationships in order to better inform the decision making process that underlies IT management. Decision making is better informed in that knowledge about the enterprise's business objectives is taken into account. 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. Building on top of this observation, MbC adds the intuition that knowledge about one enterprise's objectives is captured in - and can be extracted from - contracts and service level agreements (SLAs) that were negotiated by the enterprise. In a nutshell, MbC is about assessing available options for IT management in light of the business impact that they entail, as derived from contracts and SLAs. Available management options are compared with one another and ranked according to expected utility measures. The important thing to notice is that it is not always the case that the most obvious management option to consider is the one that would maximize the utility for the business.

Motivating 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 an IT hosting infrastructure for an electronic outsourcing company EOS. As depicted in Figure 1, EOS hosts services from two companies: KPU and EDF. Both services are web-based and require Web Servers, Application Servers and back-end systems such as Databases, Enterprise Resource Planning Systems, etc. KPU and EDF customers access these services by connecting to the servers hosted onto EOS IT infrastructure. This infrastructure can take the form of a set of independent computers or a utility data center as well as applications managed by EOS.
Both companies, KPU and EDF, have entered a contractual relationship with EOS formalized in Service Level Agreements. EOS is responsible for hosting KPU and EDF services and provides guarantees on the performance of these services.

EOS's customers, KPU and EDF are provided with different qualities of service. These guarantees are negotiated prior to service deployment and can be renegotiated over time as EOS customer's service requirements evolve. The SLAs are defined as follows:

- **KPU SLA**:
  - **Clause 1**: Availability of the hosted service will be 99% at all time.
  - **Clause 2**: Average monthly Service latency, average time to process any customers service request over a month period, will be less than 5 ms.
  - **Clause 3**: If EOS fails to meet the availability guarantee, KPU will be credited 10% of the monthly charge for the next pay period.
  - **Clause 4**: If EOS fails to meet the service latency guarantee, KPU will be granted the right to claim a 5% credit of the period over which the breach occurred.

- **EDF SLA**:
  - **Clause 1**: Availability of the hosted service will be 98% at all time.
  - **Clause 2**: Average monthly Service latency, average time to process any customers service request over a month period, will be less than 10 ms.
  - **Clause 3**: If EOS fails to meet the availability guarantee, KPU will be credited 20% of the monthly charge for the next pay period.
  - **Clause 4**: If EOS fails to meet the service latency guarantee, KPU will be granted the right to claim a 30% credit of the period over which the breach occurred.
  - **Clause 5**: Service unavailability will be fixed within 4 hours of the receipt of a trouble ticket.

Although KPU and EDF SLAs are specified over two services parameters, availability and service latency, they each define different service levels and have different penalties. In some cases, service providers segment their customer bases into three categories: Gold, Silver and Bronze and sometimes, specific Service Level Agreements are drafted with each customer depending on their needs and the service provider's ability to fulfill these needs. Regardless of the way these SLAs are established and categorized, they represent the requirements under which the service provider must deliver. As new contracts get signed between the outsourcing company and its customers, the hosting IT infrastructure is further solicited: hosted service will share servers, bandwidth, disk arrays and sometimes applications, business processes, etc.

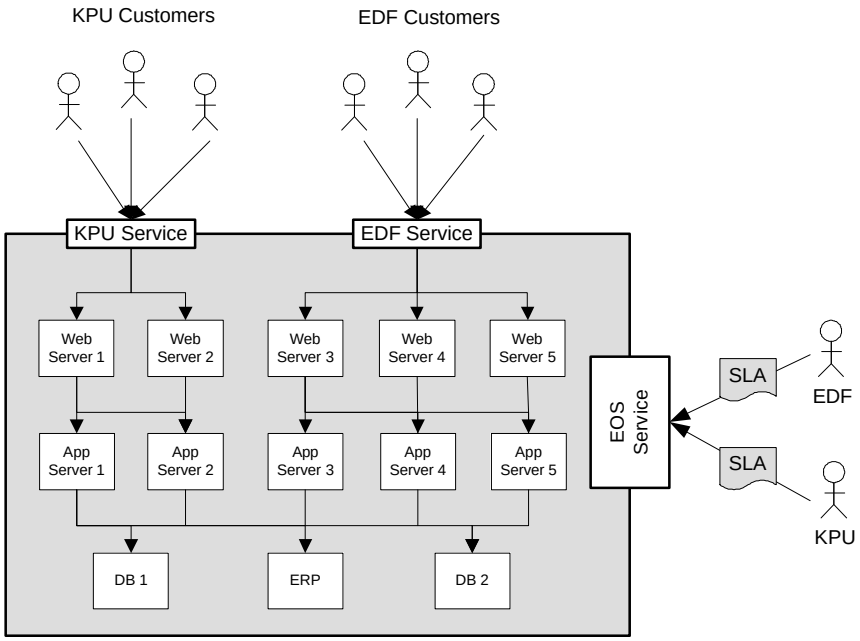

Figure 1: KPU and EDF hosted services running on EOS IT infrastructure

Let's assume for the sake of this example that the EOS hosting service exhibits a degradation of its service latency for KPU. The degradation might be the results of an under provisioning of the service resource, a failure of a web servers etc. Once the root cause of this degradation is determined, App Server 2 down for instance, the management system generates recovery plans together with expected costs, durations and impact of these strategies on service latency. These recovery plans acts on the infrastructure to restore the service to a correct state.

Let's imagine that two alternatives were identified:

- **Option1 - Repair immediately**: external consultant charging $1000 per half-day work. This is expected to result in the App Server 2 being fully down for 4 hours.
- **Option2 - Repair tomorrow, and allocate App Server 3 to KPU service**: internal charge of 500$. This is expected to result in the App Server 2 being down for 24 hours but KPU latency to be restored whilst EDF service latency would experience some degradation for 24 hours.

The Management System must then determine the effect of the different service degradation options on the current and expected service workload, and decide which of the repair options to adopt.

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.

The MbC System Architecture

The architecture of our management system extends the classic IT Management stack. At the bottom lies the *Monitoring Layer* which probes the liveliness of the service components and in 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 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 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.

The *Monitoring Layer* also provides information such as violation events and pro-active alarms (service degradation, etc.) to the *SLA State Tracking Layer* which keeps track of the current state of each SLA managed by the enterprise.

The fourth layer, *the Contract-based Analysis Layer*, informed by the SLA State Tracking Layer gives 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 associates 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.

Finally, the *Decision Making Layer* chooses the option that maximizes the utility of the service provider. The decision is then communicated to the Service and Monitoring Configuration system that applies the recovery plan associated with the decision.

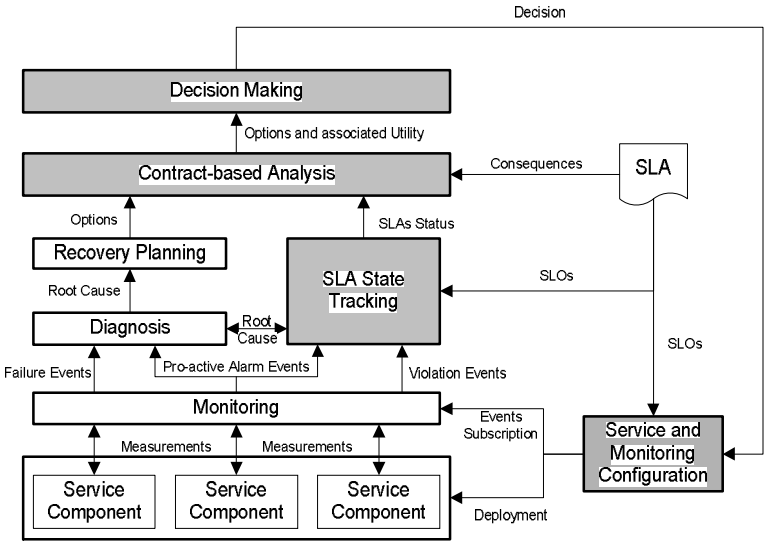

Figure 2: Business Aware Management System

Within the architecture represented in Figure 2, the components of primary concern for MbC are highlighted in gray; from the bottom up they are *Service and Monitoring Configuration*, *SLA State Tracking*, *Contract-Based Analysis* and *Decision Making*. We will describe each of these components in the remainder of this section, after presenting an overview of the underlying information model.

The MbC Information Model: The Contract Model

The MbC information model is geared towards modeling contracts and SLAs with special regard for the modeling of contractual consequences (see Figure 3). Contractual consequences are used in MBC to provide business context for the decision making process. For more details on the MbC information model, including modeling of services and resources, see [24].

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 3 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.

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.

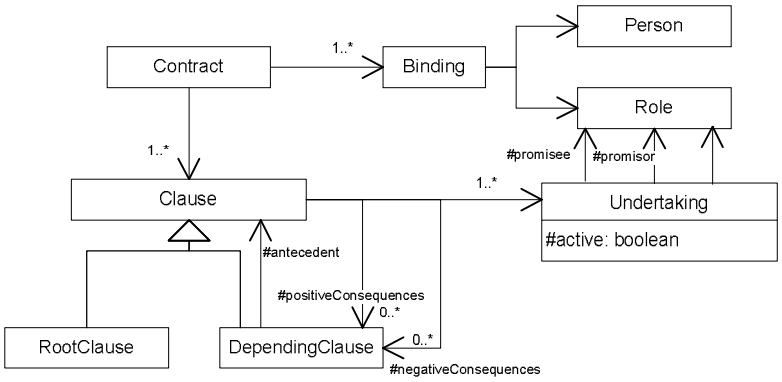

Figure 3: A view of the MbC Contract Model

As presented in the UML class diagram in Figure 3, 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 (Figure 4) 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.

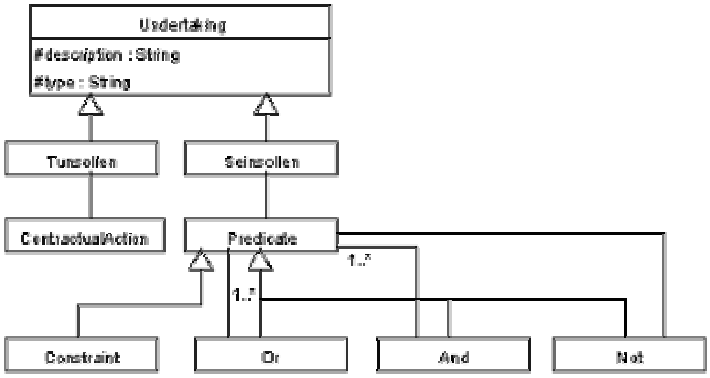

Figure 4: The Undertaking section of the MbC Contract Model

Automatic Service Configuration and Monitoring

Provisioning of the services and their monitoring is helped by information extracted from SLAs. Services are modeled as a set of parameters that describe the characteristics of the service [NOMS]. SLAs are built on parameters extracted from underlying service models and serve the purposes of deploying the appropriate service and monitoring it.

SLAs are used at deployment time to configure the services according to their description in the SLAs. For example, the SLA may explicitly specify the version of the application servers, and that the average latency of service request should be less than 5 ms. At deployment time EOS will automatically install the specified version of the application server. In addition the latency SLO is used to ensure that enough application servers are installed to deliver an average transaction latency of less than 5 ms.

After deployment SLAs are used to monitor the quality of the delivered services. At deployment time, the underlying monitoring mechanism is configured to detect changes to the service parameters that are used in the SLAs. More specifically, for each SLO in the SLAs two events are configured: a violation event generated when an SLO is violated; and a recovery event generated whenever a previously violated SLO returns back to a complying state. These events are used as described in the next section to track the state of SLAs that a service provider maintains.

In addition to violation and recovery events the service provider relies on proactive alarm events to be informed of a likely violation of an SLO. Assume that the availability SLO in the example SLAs (clause 1) is evaluated monthly. The monitoring platform constantly evaluates the availability of the service and will notify the service provider when the SLO is likely to be violated. For example, if the current availability measurement evaluates below or very close to the guaranteed value, the service provider is notified that if the current service levels are maintained a violation of the availability SLO is likely to occur. The service provider can then decide to assign more resources to the customer, assign higher priority to its requests, or take whatever other action could ensure that the violation is avoided. These options are evaluated by the decision making component described below to come up with a suggestion that takes into account the impact on the business of meeting or violating the SLAs. In some simpler cases, series of corrective actions might be triggered automatically when proactive alarms are received.

SLA State Tracking

As seen in Figure 4, undertakings associated with SLA clauses can either be active or inactive. As far as the SLA state tracking component of MbC is concerned, a state is defined by the set of active clauses that it contains. Transitions between states can either be event generated by the monitoring layer or action taken by parties in the SLA. Events can either be violation of an undertaking in the set of active clauses, fulfillment of an active clause or degradation alarms which are used to trigger corrective actions before violation of the related active clause.
For instance, Figure 5 presents a partial view of the state space of the KPU SLA given in the motivating scenario section.

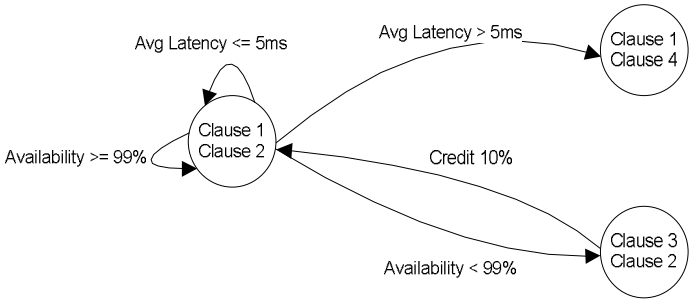

Figure 5: Partial state diagram representing KPU SLA

At first, Clauses 1 and 2 are active and the penalty clauses associated to them are inactive. The system stays in that state until violation occurs. Let imagine that the availability decreases, the monitoring layer generates an event received by the SLA State Tracking layer. As a result, the state of the KPU SLA changes to a state where Clause 2 is still active and Clause 3 is now active, note that Clause 1 is no longer active as it has been violated. Clause 3 specifies that the service providers shall credit 10% of the monthly charges to the service consumer as compensation for the breach of guarantee. Once the credit is made, the system returns to the initial state.
Of course, Figure represents only a few state of the complete state space representing the KPU SLA.

Keeping track of the state of SLAs is crucial to making decisions as it allows determining the active undertakings that needs to be taken into account in the decision process.

Contract-Based Analysis

Building on top of the state tracking system described in the previous section, the MbC contract-based analysis component performs a utility analysis of the options that are available at any point in time to the service provider, to be used as an input to the decision making process that we describe in the next section.

Contract Utility

To compute the contract utility of an option means to assess the value that an enterprise would perceive from the expected outcome of following the course of action defined in the option, given the outstanding SLAs and the current state of the system as reported by the SLA state tracking component. This kind of analysis is necessarily probabilistic as we assume with no certainty that following a course of action will result in a given outcome. Going back to the two options in our motivating scenario, there is no certainty that following option 1 will result in App Server 2 being down for *exactly* 4 hours. That is the expected downtime of the system, given with a certain value of confidence. The analysis we perform is carried out by building a decision tree that takes into account the expected outcomes of the options available to determine the likelihood of violation of the active clauses in the outstanding SLAs. The decision tree is built by looking at each SLA 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:

$$u_C(v, \lambda) = (1 - \lambda)(u_v + u_+) + \lambda u_- \quad (1)$$

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, computed recursively. Recursion stops at clauses for which the utility value can be computed in close form, such as for example penalty clauses (being the utility equaled to the inverse of the penalty cost).

Strategic Utility

Estimating the utility of an option focusing solely on the contractual utility might lead to incomplete results. The service provider might at time select to act on an option that has a lower expected utility than others. This could happen – for example - on the grounds that the service provider values not failing to deliver on promises made to strategic customers more than their expected direct utility from the SLA consequences calculated as in the sub-section above. 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. The MbC Strategic Utility model uses a heuristic rule-base to capture preferences of the service provider for quantitative strategic objectives such as "minimize the number of violations over a certain period" or "consider that violating Gold customers SLAs is three times as important as Silver SLAs".

We categorize strategic utility rules into three types:

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

For a discussion on the strategic utility model used in the current version of the MbC prototype, see [24].

Decision Making

Finally, we briefly describe the flow of information that is necessary to support the decision making process of the service provider. Figure 6 shows the high level design of the MbC decision support system. MbC presupposes a pre-processing component liable to determine the service level corresponding to a given management option to be

[1] Note that the formula given in (1) is a simplification of the general formula in that it assumes that the utility of an undertaking only depends on its violation (or compliance) and not on the entity of the violation. For example (1) could not be applied to a case where the penalty for violating an availability SLA increases with the measure of the system downtime.

executed. Service level determination depends on input on available resources, cost of repair options, service, SLAs and customers.

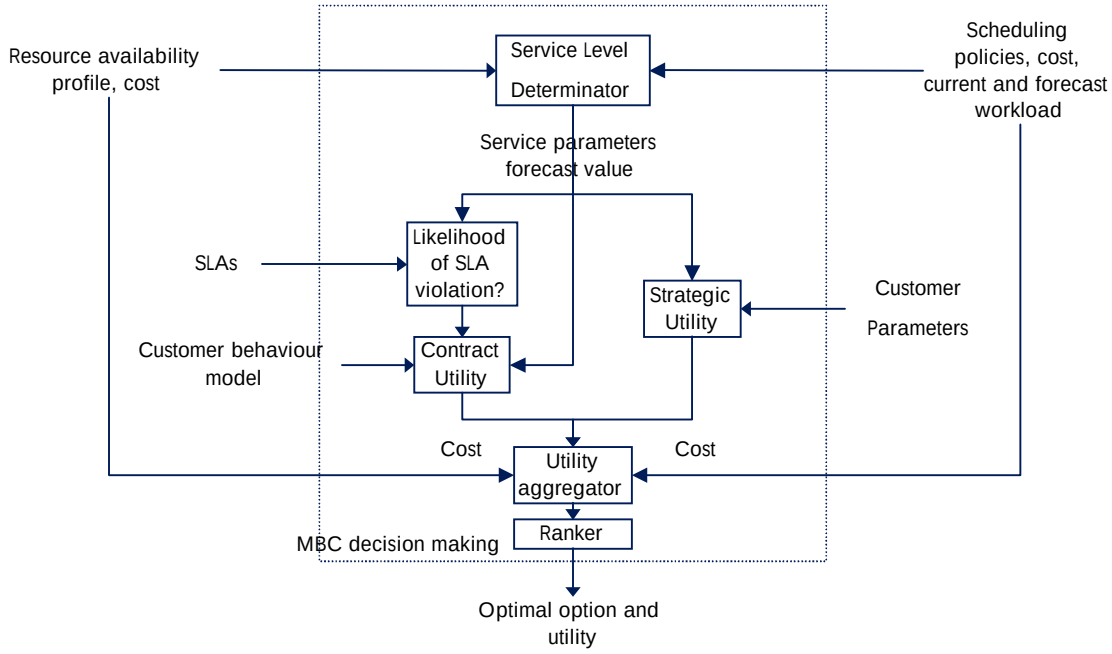

Figure 6: The MbC Decision Support System

The information on the expected service level for the selected management option is then fed onto a sub-system that determines the likelihood of SLA violation, which is in turn forwarded to the utility analysis component, described in the previous sub-section. Utility analysis is then carried out to help decide the optimal plan to follow. Together with the utility information, the engine makes use of information on the likelihood of violation of the SLAs, to come up with suggestions on the optimal management options among the ones that are available at any given time.

MbC Decision Support GUI

Through the MbC Decision Support GUI, the user is informed on the most useful management option among the ones available, based on the utility analysis explained in the previous section. The suggestion is corroborated by information on the direct cost of executing option itself; and by the loss (gain) in contractual utility that executing the option would imply because of likely compliance of violation of SLAs.

In this section we present a rendition of the current state of the MBC GUI (Figure 7). On the left hand side pane, the user is presented with the list of *issues*. An issue represents the context in which the MBC session is carried out. The system enables a user to have multiple issues under control at a given time. For any given issue, the user is presented with the list of the management options that are available, each correlated to the optimal scheduling policy. For each entry on this list, an impact monetization value is given that takes into account the three components of 1) *cost of implementation* of management options and scheduling policies[2], 2) *contract (loss in -) utility* due to the likely violation of existing SLAs and 3) *strategic utility* of the perception of the customers of the likely outcomes. The management option that corresponds to the minimum total loss in utility (*total impact*) is highlighted. All of this information is presented in tabular form. In a successive version the information will be rendered in graphical form.

The user can obtain a justification of the contract utility figures that by clicking on any of the figures in the contract utility column. The justification breaks down the total figure in the option into the contributions of the likely

[2] Scheduling policies are introduced and discussed in [24].

impacted SLAs, besides giving information on the component of the premium for meeting the SLA and a final expected utility figure that combines premium and penalties with the likelihood of meeting or not meeting the SLA.

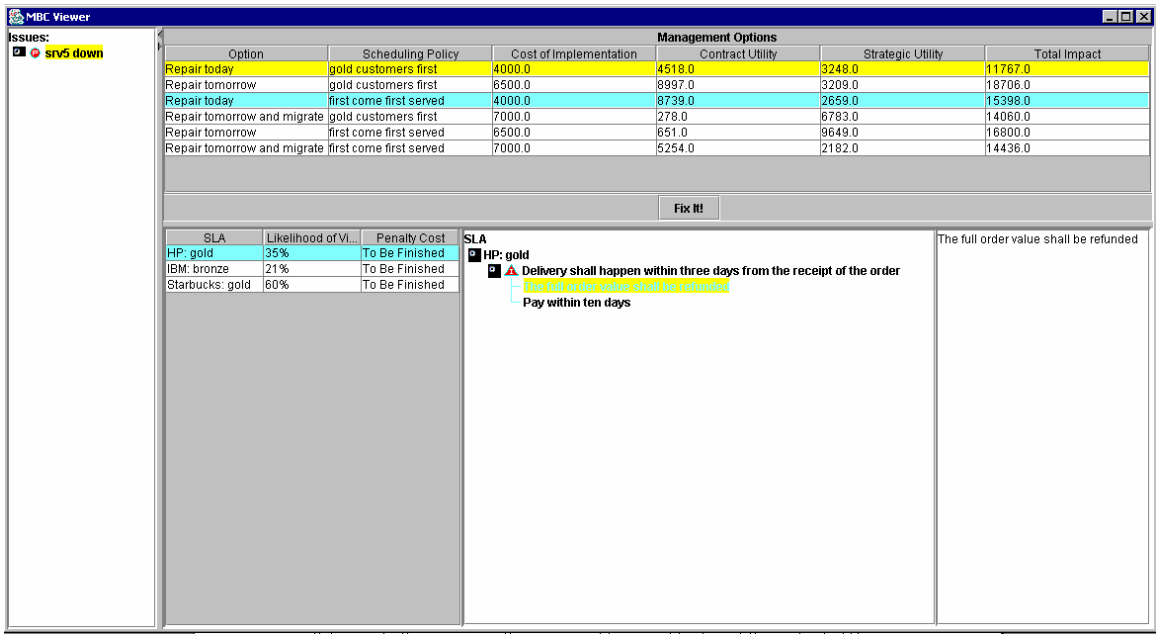

Figure 7: The MbC Decision Support System GUI

Experimental results with MbC and Future Work

Although MbC is yet to be deployed in a real-life context, we have run controlled experiments in a simulated environment built around order fulfillment management scenario. Under our assumptions (validated by experts in the order fulfillment management domain), the gain in utility from using the system was quite clear, with results well over 20 percentage points in average [24].

While the advantages of using MbC appear clear to us, we need nonetheless to understand what the tradeoff is between cost of modeling and the benefit of management by business objectives. The advantage that MbC seems to have with respect to model-based solutions is that the model itself comes for quasi-free from the formalization of existing contracts, whereas model-based systems usually require costly upfront investment in modeling before the benefits can be reaped.

The main direction in which we are currently extending our research is the formalization and extraction of strategic utility from policies and other ways of capturing the strategic objectives. We are looking into using (partially observable) markov decision processes [22] to replace our current model based on decision trees, so as to exploit the SLA state tracking model. We are also applying MbC to a frame rendering service scenario, with particular attention to the definition of penalty schemes that exhibit fairness to the service provider's customers while still ensuring that the service provider optimizes its income.

Related Work

Electronic Contracts and SLAs have been studied in various fields [1,14] 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 [5], execution [5,8,18], monitoring [7,9,12,17,18,20] 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 [21] defines a policy-based framework for management where policies are expressed with a formalism derived from event-condition-actions rules. Sloman [6] makes 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 [19]. 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 [3] 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 [15] 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.

Conclusions

We have introduced *Management by Contract* (MbC) as a paradigm to address the need for an adaptive enterprise to achieve management of its IT systems and services driven by its business objectives. We have described the architecture of a prototype MbC solution, highlighting the soft spots of our research and its future directions. MbC is yet to be deployed in a real life context, but the results that we have been obtaining, based on simulated experiments and deployment of the prototype system in a research environment look promising.

References

[1] S. Angelov and P. Grefen. B2B eContract Handling – A Survey of Projects, Papers and Standards. *University of Twente Library, CTIT Reports.*

[2] A. Avizienis, J.-C. Laprie and B. Randell, Fundamental Concepts of Dependability, Research Report N01145, LAAS-CNRS, April 2001

[3] 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

[4] F. Cristian. Understanding fault-tolerant distributed systems. *Communications of the ACM. Vol 34, issue 2, 1991*

[5] 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.*

[5] A. Daskalopulu and T. Maibaum. Towards Electronic Contract Performance. *Proceedings of International Workshop on Legal Information Systems and Applications Workshop, Dexa 2001.*

[6] 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*

[7] A. Goodchild et al. Business Contracts for B2B. *Proceedings of ISD 2000.*

[8] P. Grefen et al. CrossFlow: Cross-Organizational Workflow. Management in Dynamic Virtual Enterprises. *IEEE Data Engineering Bulletin, 24, 2001.*

[9] M. Greunz *et al.* Supporting Market Transaction through XML Contracting Containers. *Proceedings. of the Americas Conference on Information Systems (AMCIS) 2000.*
[10] B.W. Johnson. Design & analysis of fault tolerant digital systems. *Addison-Wesley Longman Publishing Co., Inc., Boston, MA, 1988*
[11] A. Johnson Jr, M. Malek. Survey of software tools for evaluating reliability, availability and serviceability. *ACM Computing Surveys. Vol 20, issue 4, 1988.*
[12] A. Keller et al. Managing Dynamic Services: A Contract Based Approach to a Conceptual Architecture. *Proceedings of NOMS 2002.*
[13] H. Lutfivya *et al.* Fault Management in Distributed Systems: A Policy-Driven Approach. *Journal of Network and System Management. Vol 8, issue 4, 2000*
[14] O. Marjanovic and Z. Milosevic. Towards Formal Modeling of e-Contracts. *Proceedings of 5th IEEE International Enterprise Distributed Object Computing Conference 2001.*
[15]D. Menascé and V.Almeida. Business-oriented Resource Management Policies for E-commerce Servers. *Performance Evaluation, September 2000.*
[16] Meta Group Worldwide Research Database, *July 2002.*
[17] A. Sahai *et al.* Automated SLA Monitoring for Web Services. *Proceedings of IEEE/IFIP DSOM 2002*
[18] M. Sallé and A.Boulmakoul. Integrated Contract Management. *Proceedings of HPOVUA Workshop 2002.*
[19] A. Schreiber *et al.* Knowledge Engineering and Management: The CommonKADS Methodology. *MIT Press(2000)* .
[20] L. Xu and M. Jeusfeld. A Concept for Monitoring of Electronic Contracts. *Infolab Technical Report Series 2003.*
[21] DMTF. *http://www.dmtf.org*
[22] A. R. Cassandra, L. P. Kaelbling, M. L. Littman. Acting Optimally in Partially Observable Stochastic Domains *Proceedings of the Twelfth National Conference on Artificial Intelligence (AAAI-94), 1994*
[23] HP. Introductory White paper to the Darwin Reference Architecture available at *http://www.hp.com/go/demandmore*.
[24] Mathias Sallé, Claudio Bartolini *Management by Contract* In Proc. 9th Network Operations Management Symposium NOMS 2004

URLs last checked at time of publication