Autonomic Self-Optimization According to Business Objectives

Sarel Aiber	Dagan Gilat	Ariel Landau	Natalia Razinkov	Aviad Sela	Segev Wasserkrug
IBM Haifa Research Lab	*IBM Haifa Research Lab*	*IBM Haifa Research Lab*	*IBM Haifa Research Lab*	*IBM Haifa Research Lab*	*IBM Haifa Research Lab*
sarel@il.ibm.com	*dagang@il.ibm.com*	*ariel@il.ibm.com*	*natali@il.ibm.com*	*sela@il.ibm.com*	*segevw@il.ibm.com*

Abstract

A central challenge in the runtime management of computing environments is the necessity to keep these environments continuously optimized. In this paper we introduce a new paradigm, which focuses on self-optimization according to high-level business objectives such as maximizing revenues. It replaces the more traditional optimizations that are based upon IT measures such as resource availability. A general, autonomous process is defined to enable such optimizations, and a set of technologies and methodologies is introduced to support the implementation of such a process. The paper concludes with two types of validation tests carried out on an eCommerce site, that demonstrate the value and applicability of this approach.

1. Introduction

One of the major challenges in the runtime management of computing environments in any enterprise is both the initial optimization of these environments, and keeping these environments optimized when changes occur. Current goals for such optimizations typically focus on IT measures (e.g. increase the site's availability by 1%, reduce overall response times to an average of 3 seconds). Nowadays, however, many enterprises are keen on satisfying business objectives, such as maximizing the total income generated by the infrastructure. Therefore, IT optimization should focus on these business goals, rather than the more conventional IT metrics.

Optimizing the IT infrastructure according to such business objectives is not a trivial task, as it is unclear how settings of parameters at the IT level will affect the business objectives. Moreover, as stated above, optimizing the IT infrastructure is not a one time effort. This is due to the fact that there may be changes which occur in the environment in which the infrastructure operates, that may render any pre-defined setting suboptimal. Two examples of such changes are failures of hardware and software components, and significant changes in the mix or load of users of this infrastructure. Thus, the solution requires both (1) an automatic mechanism for carrying out IT optimization according to business metrics, and (2) an automatic mechanism for recognizing significant changes and re-optimizing the system as a result of such changes, enabling what we term on demand self-optimization.

In this paper, we present a general architecture and a set of technologies and methodologies enabling such an on-demand optimization, as well as two case studies demonstrating the applicability of this approach.

The remainder of this paper is organized as follows: Section 2 reviews existing IT optimization techniques, i.e. techniques that focus on optimizing IT level metrics. Section 3 describes a general process for self-optimization according to business objectives. Sections 4 and 5 describe the technologies and methodologies used to implement this process. Section 6 describes validation tests carried out regarding the methodologies described in sections 4 and 5. Finally, the article concludes with a summary in section 7.

2. Related works

Most current IT related runtime optimization efforts focus on optimizing IT level metrics, such as overall application response times (e.g. [9]), or component specific (e.g. DBMS) response times. However, these IT metrics are not directly related to business level

Proceedings of the International Conference on Autonomic Computing (ICAC'04)
0-7695-2114-2/04 $20.00 © 2004 IEEE

objectives, such as profit or return on investment (ROI). Some works and products attempt to manage IT so as to better align it with business level objectives. Examples of this include: The Peakstone eAssurance product ([11]), which attempts to manage a web site so that a set of service level agreements is not violated and [8], which attempts to maximize income of an eCommerce site by prioritizing the customer's requests based on the customer history and current shopping cart status. However, we are unaware of any works in which a general approach, architecture or technologies are introduced that allow IT optimization according to high level business objectives.

3. Business objective driven optimization

We define a general automatic optimization process by which such objective driven optimization can be carried out. Figure 0 depicts this process.

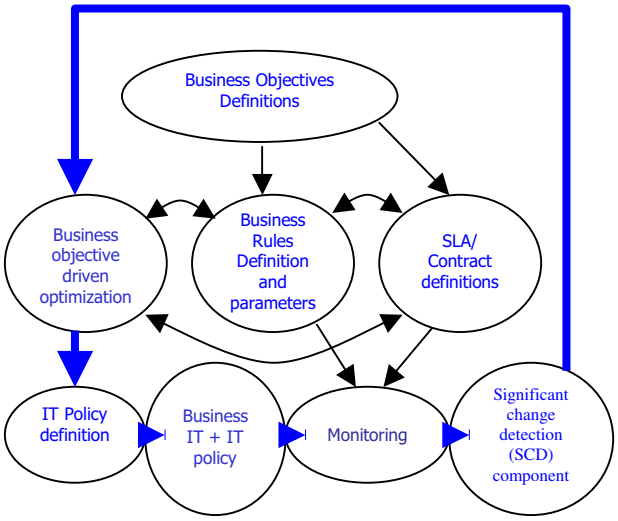

Figure 1: Business objective driven optimization Process

To start the process, business objectives are defined. Based on these definitions, both business rules detailing how IT level metrics affect this business objectives, and service level agreements (or other contracts), are generated. Once the business rules and SLA/contracts are defined, the following continuous loop is executed: Business objective driven optimization is carried out; the IT policy defined by the optimization process is passed to the runtime environment to control the IT infrastructure; monitoring tools present in the environment monitor the IT resources, the compliance with the contract definitions, and the business objectives achieved by the infrastructure; the IT business infrastructure is constantly monitored by monitoring components in the environment; the Significant Change Detection (SCD) uses the results of the monitoring process to decide whether a significant enough change exists between the model and the business environment; if a significant change is detected, the model used for the business objective driven optimization is updated, (resulting in on-demand optimization), and the loop begins again starting with the optimization process.

The process defined above is a general one, and many components and models can be used to implement this process. In Section 4, technologies implementing such a process are described in more detail.

4. Implementation technologies and methodologies

This section describes a set of technologies and methodologies used to implement the process outlined in Section 3. The components of this process detailed in this section are: business rule definitions, business objective driven optimization, IT policy definition and management, monitoring, and significant change detection.

4.1. Business rule definitions

The business rules may be expressed in a variety of methods, e.g. general economic models and rule engines. In our current implementation of this process, these rules are expressed using AMIT (Active Middleware Integration – described in [1]).

AMIT is both a rule language for specifying situations – a complex temporal predicate on events, and an efficient event correlation engine for detecting the occurrence of these situations. The definition of situations is recursive, in that situations can then be used in turn as input events to other situations. This allows for hierarchical specification and calculation of even more complex event predicates. The AMIT rule language can be used in many different applications and various domains, as an event in the context of this language can be anything from a basic IT event to a complex business process event. Examples of such events are: the failure of a disk, a measurement of the load on a web server, data regarding a specific client web request, and the status of a business process.

Examples of business rules, expressible in AMIT, include "For each buying or selling of stock, a 4% commission is earned", and "For each service level

agreement violation, a $50 penalty is paid". More detailed examples of business rules, as well as a sample rule defined in the AMIT language, appear in Section 6.1.

4.2 Business objective driven optimization

An implementation of business objective driven optimization requires a model of the system. In our implementation, this model is composed of three main sub-models: A business level model, an IT model and an IT to business level impact analysis model.

The business level model supports the calculation of the business metrics. This may include factors such as gains from commissions, explicit penalties paid to customers whenever service level guarantees are violated, customers deserting due to poor service, gaining new customers due to good reputation, and losing customers due to poor reputation. In order to enable optimization, this economic model should also enable the calculation of an "end result" – a single quantity that can be used to quantify the alignment of the IT with the business objectives. An example of such a quantity is the total income generated by the IT infrastructure. We term this quantity the *overall business metric* (OBM).

The IT model is composed of the system model and the system user behavior model. The system model takes into account the hardware configuration of the IT – e.g. number of servers, number of CPUs on each server, network configuration, and the software, which includes the applications supported, the behavior of these applications and the resources required by each application. The system user behavior model takes into account the manner in which the users of the IT infrastructure use the systems supported by this infrastructure.

The IT to business level impact analysis model defines how events at the IT level impact the business objectives defined by the business level model. Examples of this are how individual response times measured for a customer impact the penalty paid to a customer and the measurement of a customer's dissatisfaction with poor response times.

In our current implementation, the system user behavior model and the system model are modeled using hybrid simulation models (i.e. simulation models combining both conventional detailed simulation modeling techniques), and other techniques. Additional details regarding these models appear in Section 5.

The mechanism for expressing both the impact of the IT level rules on the business objectives and the calculation of the overall business objectives is the same AMIT technology used for specifying the business rules. Moreover, the rules used for calculating these quantities are a superset of the business rules definitions and the SLA/contract definitions, as these rules must be taken into account when calculating the overall business objective (OBM). In addition, the IT policy tools present in the environment are also modeled, in order to enable the prediction of the effect of changing these policies on both the infrastructure and the business objectives.

In order to optimize the IT according to the business objectives, an optimizer is coupled with the model. The optimization process itself is a search over the space of possible actions/policies, which attempts to find the setting in this space that optimizes the business objectives. A more precise description of the optimization process appears in Figure 2. Figure 3 depicts the architecture of the optimization mechanism.

```
// Optimization loop
While (true)
{
–  Optimizer generates policy A
–  Simulate model to calculate OBM
   for A
–  If new OBM significantly better
   than OBM for existing policy,
   set real system policy to A
–  Provide OBM result as input to
   optimizer
}
```

Figure 2: Optimization algorithm

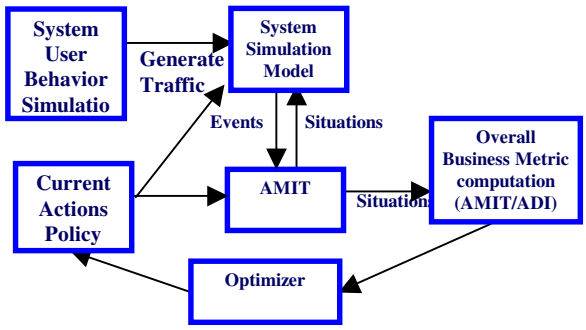

Figure 3: ARAD architecture

Currently, Tabu Search [4] is the optimization algorithm used. However, any black box optimization algorithm can be utilized.

4.3 IT policy definition and management

In order to affect the production environment, an *effector* must be present that can manage the environment's IT resources according to a specified policy. Moreover, different settings of this policy must

have a significant impact on the business objectives and the OBM. In an e-Commerce web site, for instance, if the purpose is to maximize profits by giving better service times to customers that bring higher value to the site, an effector must be in place, enabling the differentiation of the resources given to each customer.

An example of such an effector is the bandwidth allocation component of IBM's Edge Server's TQoS component ([13]). This component allows allocating the network bandwidth of user's HTTP requests, thereby allowing to prioritize the requests of one customer over another. This bandwidth allocation is carried out using a *token bucket flow* algorithm ([5]).

An example of an edge server policy appears in Figure 4. From this example, it can be seen that even though this effector enables giving one group of customers a higher priority than another, it is not clear how to set the values for this policy so as to optimize high level business objectives.

```
policyAction PlatinumHigh
{
     PolicyScope  DataTraffic
     DiffServExcessTrafficTreatment drop
     DiffServInProfileRate         500
     DiffServInProfileTokenBucket   1000
}

policyAction GoldHigh
{
     PolicyScope  DataTraffic
     DiffServExcessTrafficTreatment drop
     DiffServInProfileRate          300
     DiffServInProfileTokenBucket   1000
}
```

Figure 4: Bandwidth allocation policy example

4.4 Monitoring

The optimized policy generated may be rendered sub-optimal by changes in the system environment. Therefore, this environment must be constantly monitored in order to detect such changes. In the technology described in this article, the AMIT runtime engine, described above, enables the monitoring of the business results in the real environment.

4.5 Significant change detection

In our implementation, there are two cases in which a significant change is detected. The first is the case in which any event that is defined as an event that signals when a significant change occurs. An example of such an event is the failure of a server. The second case is as follows: As the entire process is business objectives driven, a significant change is defined as a significant deviation of the monitored business objectives from the business objectives predicted by the models of the optimization component. Statistical methods are used to detect such a significant deviation. In our current implementation, this is carried out by the following general procedure: Several key business metrics are chosen to be tested for a significant change, and the results for these metrics in the actual environment are constantly compared with the results for these metrics in the simulated environment, using the the χ^2 statistical test (([2]). A more detailed description of our implementation follows:

- The key business metrics that are to be tested for a significant change are chosen.
- The time line is divided into *periods*, each containing *n intervals*. A sliding window of change detection is defined, consisting of *k* periods.
- As soon as a period is over in the real environment, *k*n samples* are collected for each metric. A sample for an accumulating metric (such as the total income) is calculated as the difference between its value at the end and the beginning of an interval. A sample for an averaging metric (such as the average response time for a customer) is calculated as the average of the metric over the interval.
- An analogous set of sample points is then collected from the predictive model (using several simulation runs), for the same window of time periods under inspection. The number of simulation runs is determined so as to ensure that an accurate enough estimator of these metrics is generated.
- The χ^2 statistical test is used to test whether the observed distribution (the samples from the real environment) is a good fit to the expected distribution (the samples from the simulated runs). Such a test is performed for each of the key business metrics. A false result of the χ^2 test for any of the metrics indicates the occurrence of a significant change in the environment.

A concise description of the above algorithm appears in Figure 5.

5. Modeling

In order for such on-demand optimization to be widely applicable, two modeling related issues must be resolved: The first is the ability to create the simulation models in a simple, non-programmatic manner. The second is the automation of the model updates, required to enable the correct handling of changes in

the environment. The following sub-sections describe the methodologies used to address these issues.

```
// Significance Change Detection
At the end of a period, for each
business metric to be tested:
{
-  Calculate the sample points for
   the business metric from the
   real environment (n samples for
   each of the last k periods)
-  Do the same for the simulation
   runs
-  Perform the χ² test for the real
   environments vs the simulated
   runs
-  If the test returns false,
   signal a significant change
}
```

Figure 5: Significant change detection algorithm

5.1 Model creation

To enable model creation in a simple, non-programmatic manner (given an IT site), a two stage approach is used:

1. The standard components of the IT (e.g. middleware, operating systems and hardware) are modeled using a set of pre-prepared building blocks. In this approach, for each type of component, a model is created in advance, which can be re-used across different sites and infrastructures.
2. The site specific models, (e.g. application, user behavior) are created automatically using machine learning and statistical techniques.

As the purpose of our technology is to optimize business objectives, there may be cases in which a detailed model of a specific middleware or hardware may not be required. Therefore, there can be several types of building blocks. One example of a building block type is one which contains a detailed queuing model. However, there may also be other types of building blocks, such as a "black box" building block. Such a building block is simply a function, *f*, that for each incoming request to the building block, returns the response time of that request. More detailed examples of specific building blocks appear in Section 6.1.

With regards to the site specific models, the following models and learning algorithms have been defined by us for the domain of e-Commerce web applications:

- **User behavior model:** This is a model of the traffic pattern of the requests that enter a web site. This traffic pattern is modeled using a combination of the following: A set of Markovian state-transition graphs (the CBMG model [10]), and a set of clusters. Each Markovian state transition graph represents a possible web session type, with each node in the graph representing a possible request. The set of clusters models the frequency with which each type of user session is initiated by a user. Each simulated user is then assigned to a cluster, to determine the frequency with which this user initiates each type of session.

  These models are automatically derived using the following techniques: The Markovian graphs are derived from web logs using the CBMG derivation algorithm ([10]), and the clusters are derived from the web logs using the k-means clustering algorithm ([3]).
- **User Attribute Model:** This models the values of the attributes important to the business rules (such as the purchase amount in a purchase request), that are a part of the above HTTP requests. These attribute values are modeled using a set of probability distributions. These distributions are automatically generated from logs and database values on the customer's site, using statistical methods.
- **Tier level message breakdown:** This models the breakdown of incoming user HTTP requests (modeled by the user behavior model), into invocations of different components of a multi-tier web application, e.g. Servlets, JSPs, EJBs and database requests. For each such HTTP request, the tier-level message breakdown is modeled by a probabilistic graph, whose nodes are services and methods that are invoked.

  The above Tier-level message breakdown graphs are built by applying business-process analysis ([7]) and distribution fitting ([2]) algorithms to a log of broken-down client requests produced by tracing and monitoring tools.
- **Web and Application Server resource requirements:** These are models of the resource level (e.g. CPU, disk) service times (i.e. the net usage time of each resource) of each web and application level request. The service time is modeled by a load dependant distribution function. These functions are automatically generated using statistical and machine learning techniques using data gathered from performance tools at the web site.

5.2 Model updates

Whenever a significant change is detected in the system, the model must be updated to ensure that reliable results are obtained by the optimization processes. Therefore, automatic procedures are required for updating the model.

Updates may be required either in the standard components of the system (e.g. as a result of the failure of a server), or in the site specific models. In the case of updates to the standard components, the model is modified by the addition, deletion, or update to a building block. For example, if a server fails, the building blocks representing that server and the middleware executing on that server are removed from the model.

Updates to the site specific models are handled, (similarly to the initial derivation) by using machine learning and statistical techniques. Following are more details regarding such an algorithm for the update to the system-user behavior model.

In order to update this model in response to a significant change in the environment, a copy of the client traffic model is constantly updated from monitored system traffic, using on-line clustering algorithms. Whenever a significant change is detected, the model is incorporated into the predictive model, and the optimization process restarts with the new model in place. Figure 6 outlines the algorithm.

```
// Model Update
- Make a copy of the user
  attribute model
- While no significance change is
  detected:
{
  Update the copy of the model
  with real environment input,
  using online clustering methods
}

- Once a significant change is
  detected substitute the copy for
  the model in the predictive
  system
```

Figure 6: Automatic model update algorithm

6. Validation

Two main types of validation tests were carried out by us. The first is optimizing an e-Commerce site according to business objectives, intended to test whether carrying out the above outlined process can indeed result in a significant improvement to business results. The second is recognizing significant changes in such a site, and re-optimizing accordingly, intended to test the value of carrying out the optimization process each time a significant change occurs.

A detailed description of optimizing a web site appears in Section 6.1. Regarding tests of the second type, a detailed description of recognizing and reacting to significant changes is beyond the scope of this paper. However, following is a brief description of such a test:

- An e-Trading site's policy was optimized according to a certain number of users (10,000).
- The number of users was increased significantly (15,000), without changing any of the other model parameters.
- Such a change was recognized and re-optimization was carried out.
- The policy after the re-optimization yielded a significant improvement in business results (more then 23%) over remaining with the previous policy.

6.1 Validation of the optimization architecture

In this validation test, an e-Trading web site (i.e. a web site allowing the buying and selling of stocks on the internet) was optimized according to business objectives. This e-Trading web site was based on IBM's Trade3 benchmark application ([14]), i.e. a three-tier J2EE e-Trading web application.

The following steps were carried out as a part of this test:

- The site's architecture was defined. This architecture included the following components:

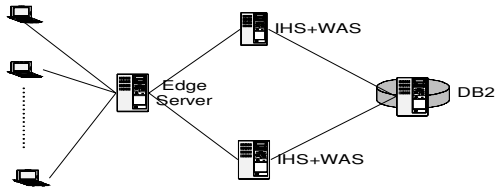

Figure 7: Case study architecture

  - Two web/application servers, i.e. two servers, so that on each server, both IBM's HTTP Server (IHS) and IBM's Websphere Application Server (WAS) are installed.
  - One DB server, using IBM's DB2 that serves the database requests from both web/application servers.
  - An Edge server, which serves as the effector. In this configuration, the Edge Server not only allocates the bandwidth, but serves as a load balancer, i.e. it directs incoming web requests to actual web servers.

A diagram of this architecture appears in Figure 7.

- The business objectives were defined as follows:
  - For each buying/selling of a stock, a commission of 4% or $25 is collected, whichever is greater
  - Two types of service level agreements (SLAs) exist, Platinum and Gold. A platinum customer pays a daily flat fee of $50 and a gold customer pays a daily flat fee of $20.
  - A platinum customer is promised an average response time of 1 second. For each 1% deviation from this response time, the site pays a penalty of $5.5.
  - A gold customer is promised an average response time of 2 seconds. For each 1% deviation from this response time, the site pays a penalty of $3.5.
- These business rules were then written as AMIT rules. An example of an AMIT rule that is used for the calculation of the commission appears in Figure 8.

```
<situation internal="true"
  lifespan="NeverEnding"
   name="AggregatedComission"
   persistent="false">
    <operator>
         <report detectionMode="immediate"
          repeatMode="always" where="">
        <operandReport addToSum="amount!=0"
          average=""
          event="Transaction" max="" min=""
          override="false"
          partAvg="false" partMax="false"
          partMin="false"
          quantifier="each" retain="true"

sampleMeasurementUnit="occurrences"
          sampleRate="1"
          sum="max(25,amount*0.04)"
               threshold=""
             />
        <operandReport addToSum="true"
               average=""
          event="DayStart" max="" min=""
          override="false"
          partAvg="false" partMax="false"
          partMin="false"
          quantifier="each" retain="true"

sampleMeasurementUnit="occurrences"
               sampleRate="1"
               sum="0" threshold=""
             />
         </report>
      </operator>
      <situationAttribute expression="sum"
      name="commission" type="number"/>
      <comment commentText=""/>
  </situation>
```

Figure 8: Example of an AMIT rule

- The architecture was modelled using building blocks. The following building blocks were used:
  - A hardware building block: This is a queuing network building block, which contains queues and servers for physical resources such as CPUs and disk.
  - The IHS and WAS server: These were queuing network building blocks. The IHS model enables taking into account parameters such as connection pooling and web server process pool, while the WAS model consists of several components such as web container, EJB container and JDBC pool.
  - A black box model of DB2 was created, i.e. the DB2 is represented by two functions. The first function, $\underline{r} = f\left(R_i, \underline{\theta}\right)$ returns the response time for request R_i given the state of the database, $\underline{\theta}$. The second function, $\underline{\theta} = g\left(R_i, \underline{\theta}'\right)$, returns the new state, $\underline{\theta}$, given the new request R_i, and the previous state, $\underline{\theta}'$. The state encapsulates all factors that affect the response time of the request, such as the CPU load on the database server.

    In our implementation, Bayesian Network Learning algorithms are used to generate these functions from performance measurements in the environment.
  - The model was created based on these building blocks, using XJTek's ([12]) Anylogic 4.5 simulation product. A diagram of the simulation model appears in Figure 9.

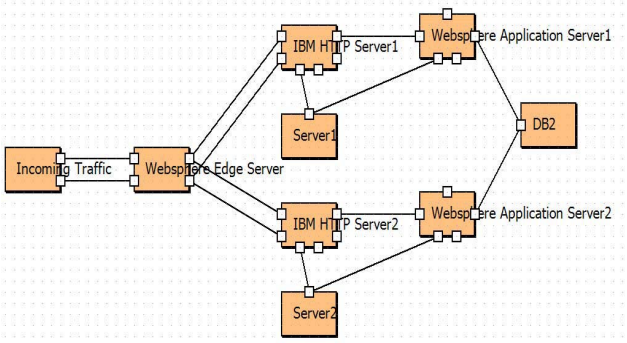

Figure 9: Simulation model

- The edge server policy was optimized using the Tabu Search algorithm.

The results of the case study were the following: With no effector policy (no bandwidth limits), the total income was -$475,376. The optimization process, after 7 iterations, improved the total income and brought it to $179,088, i.e. changed an initial negative income to a positive value. Figure 10 shows these results.

6.2 Conclusions

Based on the above two validation tests, the following conclusions can be reached:

- Given a valid model of the system, very significant improvements in business objectives can be obtained using this methodology, by setting the policy of currently available tools such as IBM's Edge Server.
- In order to keep the site optimized according to business objectives, it is important to recognize significant changes and respond to them.

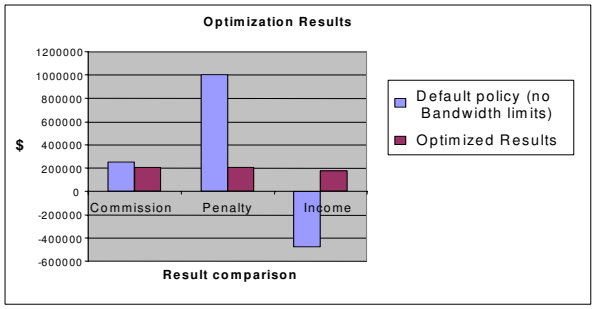

Figure 10: Default vs. optimized policies

7. Summary

This paper presented a process, architecture, and a set of technologies for autonomic, on demand optimization of an IT business infrastructure according to high-level business objectives, rather than IT level metrics. This approach has benefits of constantly keeping the infrastructure aligned with business objectives, and results in a clear connection of IT related policy decisions to business level metrics such as profit or ROI.

The architecture of the optimization component is very general, and may apply to various IT scenarios, such as e-Commerce sites and messaging infrastructure. Although simulation technologies were used in the implementation presented in this article, other types of models, such as functional and analytical models, may be incorporated into this architecture. In addition, it is possible to replace the rule-based economical models presented in this article with other types of models.

Several future research directions are available. Most implementation choices for the different algorithms and techniques are not exclusive. Analytical and functional predictive models should be examined and analyzed. The definition of suitable economical models should be explored. The Tabu-search algorithm used for the optimization process could be substituted with another. Finally, the various ways in which to automatically acquire various models should be explored.

8. References

[1] A. Adi and O. Etzion, "The Situation Manager Rule Language", *International Workshop on Rule Markup Languages for Business Rules on the Semantic Web*, 2002, pp. 36-57.

[2] Brownlee, K. A., *Statistical Theory and Methodology in Science and Engineering 2nd Edition*, Robert E. Krieger Publishing Company, Inc., 1965.

[3] Everit, B., *Cluster Analysis*, Halted Press, New York, 1980.

[4] Glover, F. and M. Laguna, *Tabu Search*, Kluwer Academic Publishers, 2002.

[5] L. Huynh, "Policy Based Quality of Service in z/OS V1R2", IBM Corp., 2001, http://www-1.ibm.com/servers/eserver/zseries/library/techpapers/pdf/gm130027.pdf

[6] Z. Ghahramani, "Learning Dynamic Bayesian Networks", *Adaptive Processing of Sequences and Data Structures* . Lecture Notes in Artificial Intelligence, pp. 168-197.

[7] M. Golani and S. Pinter, Generating a Process Model from a Process Audit Log, Business Process Management 2003, pp. 136-151

[8] D. Menasce, J. Almeida, R. Fonseca, and M. Mendes, "Business-oriented resource management policies for ecommerce servers", Performance Evaluation", 2000, 42(2-3): pp. 223- 239.

[9] D. Menasce, D. Barbara, and R. Dodge, "Preserving QoS of E-commerce Sites Through Self-Tuning: A Performance Model Approach", Proceedings of 2001 ACM Conference on E-commerce, Tampa, 2001, pp. 224–234.

[10] D. Menasce, V. Almeida, R. Fonseca, and M. Mendes, "A Methodology for Workload Characterization of E-commerce Sites", Proc. 1999 ACM Conference in Electronic Commerce, Denver, CO, Nov. 1999.

[11] http://www.peakstone.com

[12] http://www.xjtek.com

[13] http://www.ibm.com/software/webservers/edgeserver/

[14]http://www-3.ibm.com/software/webservers/appserv/benchmark3.html