ITBVM: IT Business Value Modeler

Nikolai Joukov, Birgit Pfitzmann, HariGovind V. Ramasamy, Norbert G. Vogl, Murthy V. Devarakonda, and Tryg Ager
IBM Research, Hawthorne, NY, and Almaden, CA, USA

Abstract

Today, enterprise IT environments are complex as never before with individual applications, tiers, or technologies segregated into individual management domains. Typically, the value of business applications and the dependencies between business and IT objects and IT objects among each other is unknown or at least not up to date. Thus, ultimately, the business value of individual IT tasks is unknown. Hence it is very hard to perform global management services such as performance optimization in resource-constrained environments. This deficiency is even more deeply felt by an internal or external services provider called in to perform an optimization or to improve an IT management framework.

We propose a framework ITBVM for business-value driven IT optimization with particular emphasis on such enterprise environments. A key part is the use of discovery technologies to provide the link between business value and IT objects. As one instance of the framework, we show how discovery can improve a performance-optimization problem in an otherwise blackbox scenario. We validate these improvements through experiments in a controlled setup and through statistical interpretation of fine-grained dependency discovery in a large real enterprise environment.

1 Introduction

We address the problem of business-value based IT optimization in large enterprise scenarios as typically encountered by services providers. In particular, we will show the importance of using discovery technologies to link IT objects and IT tasks with business values, and we will demonstrate how the discovered knowledge can aid global algorithmic value optimization.

Today, many large businesses and government organizations have a diverse body of data centers running many generations of servers and software and serving diverse business functions or missions. IT infrastructure is commonly built up over decades and is the result of complex mergers of various smaller IT environments. Ad-hoc management of IT in enterprises leads to under-provisioning, over-provisioning, or usually both, and to sub-optimal decisions on task admission control when resources are insufficient for all requests. The problem of optimizing IT environments according to the business needs is important, yet largely unsolved. A key challenge is that significant disconnect exists between business management and IT management. Given the heterogeneous nature of modern data centers, and IT design and management segregation into smaller domains, IT managers usually do not precisely know what a particular piece of software or hardware is used for, nor how the pieces are connected in terms of data dependencies, network connectivity, module dependencies, etc. While architectural principles and trends like autonomic computing, Services-Oriented Architectures, and now Cloud Computing strive towards modularizing enterprises into well-documented components and orchestration of the usage of these components, in most real enterprises they only govern a very small part of the IT environment, and even then the documentation is usually not complete.

If an internal or external services provider is asked to transform and optimize the IT in an enterprise beyond individual small management domains, the lack of knowledge about business-to-IT and IT-to-IT dependencies is even more strongly felt. Hence dependency discovery is becoming an important sub-service for many engagements.

1. Quantify the value of business activity
2. Map value network entities and flows to IT entry points
3. Discover end-to-end dependencies between IT infrastructure, software, and data

Figure 1. IT optimization based on a business model expressed as a value network.
In this paper, we present a framework for expressing business-level optimization criteria, making the connections between these criteria and IT objects, and performing automatic optimization deep into the IT stacks and tiers according to business value. For example, Figure 1 shows (1) an enterprise model based on value networks, which may include intangible value exchanges, (2) connection to the IT level via IT entry points, and (3) discovery of IT dependencies, within stacks and across servers and tiers.

Global optimization for maximum value, our goal, is more complex than prioritizing different tasks or objects. For instance, if each execution of a business task \( T \) has value \( x \) and each execution of business task \( T' \) has value \( 2x \), but \( T \) and \( T' \) compete for utilization of a critical resource \( R \) and task \( T' \) needs 3 times as much of \( R \) as task \( T \), it is better to give task \( T \) priority. Whether there is indeed a bottleneck on resource \( R \), and what utilization each task consumes there, is usually not known in advance and will be implicitly determined in the global optimization.

We have built a system called IT Business Value Management (ITBVM) that allows expressing business-level optimization criteria and constraints using simple mathematical expressions. We use Galapagos [12] to discover and express IT-level dependencies. Galapagos is a system and methodology for automated discovery of fine-grained dependency information between applications, middleware, and data. We demonstrate the capabilities of this system for efficient optimization of IT resources in two enterprise environments.

## 2 Business Value-driven IT Optimization

It is desirable to manage overall IT environments, and not only individual servers, in order to ensure that resources are spent on the most valuable tasks first. The ultimate decision criterion should be business value or similar global criteria in not-for-profit organizations. In this section, we describe how one can relate business-level management goals with low-level IT management, in particular in services engagements for realistic heterogeneous, not fully documented enterprise environments.

### 2.1 Business Value

In order to maximize the business value of IT operation, one first needs to be able to quantify business value. For major entities of an enterprise, business value may be known from balance sheets or from detailed business processes annotated with financial information, see, e.g., [4]. For tasks not directly associated to business value generation, value networks can model the flow and values of intangible assets between business or organizational entities [16, 27]. If no direct financial value can be given for certain business tasks, one may assign perceived value based on user preferences or feedback. This is a typical case in government organizations and in functional units of enterprises where no value network analysis has been made, e.g., Human Resources or Research. In IT-related services engagements, the client may provide known values of business activities to the services provider, or the services provider may help the client find them out, or the client only desires a framework where they can later input current values themselves.

### 2.2 Bridging Business and IT: IT Entry Points

Business values and goals are usually expressed at the business level in business-level terms. Most IT management decisions are not easy to relate to the affected business goals and values. Hence even where business values are known, they are typically not used in IT optimization, or only in a very coarse way by assigning priorities or overall IT budgets to lines of business.

We use IT entry points to map business value to IT. An IT entry point is a means by which a value network entity (usually, a human, an external contact, or a machinery) accesses IT. Typically, complete documentation about entry points does not exist. However, since value network entities directly use these entry points, the entry points can be found by interviewing appropriate people in the organization. An entry point is often a set of URLs.

An IT entry point may correspond to inputs or outputs of an IT system. Value is typically associated with generated outputs. The outputs may occur at the same IT entry points as the inputs, e.g., in web requests, or at different IT entry points, e.g., in stream processing [25]. Hence, even if the control mechanism used for optimization is admission control, one cannot simply give priority to the highest-value entities but has to perform global optimization for the resulting outputs as described in the example in Section 1.

Semi-automated techniques may support the mapping of business value to IT entry points. For example, access logs may show which users and thus which business roles accessed which URLs, and off-the-shelf software such as a financial application may be associated with the business unit it is typically used for.

### 2.3 Discovering and Leveraging IT Dependencies

Once value network entities and thus values have been mapped to IT entry points, we need to understand the dependencies that each IT entry point has with other IT objects. Typically, this step is far more difficult than identifying the entry points: Manual mapping of business values to IT entry points is possible because it is directly related
In the end-to-end analysis of these dependencies, we see the activity of people using IT for some business reason. However, it is infeasible or at least prohibitively expensive to manually track all the complicated dependencies between IT components in a large enterprise, because the number of components and links is large and the time of skilled IT personnel per server is limited. We saw servers that are involved in hundreds of business applications, while each system administrator manages dozens of servers and rarely knows about servers, stack levels, or tiers outside of their management domain. Shared objects imply that even after discovery one cannot simply modularize the IT optimization task into one top-level optimization among the business applications, followed by one internal optimization per business application.

Figure 2 shows our use of the fine-grained application discovery capabilities of Galapagos [12] to link a business-level value network with the IT infrastructure objects. The figure shows IT entry points for value network actors, typically human users, in the form of URLs denoting web objects. Other entry points such as rich client applications or external systems connected by other protocols are also conceivable. The example shows two value network actors. One uses URLs corresponding to web objects WO1 and WO2, the other uses a URL corresponding to WO3. While WO2 is a static web page, the other two URLs are configured to lead to applications AP1 and AP2 on an application server AS1. At the next tier, we assume we discovered that application AP1 depends on database D1, and application AP2 on database D2. The grey outer objects denote the servers where these applications reside.

In the end-to-end analysis of these dependencies, we see that VN Actor 1 does not depend on database D2 and VN Actor 2 does not depend on database D1. This fact would not have been revealed by a pure server-to-server dependency analysis or even network-based discovery with port-level details. The latter would distinguish as far as the services, which we show as an outer white box in each case, i.e., in this example web servers, application servers, and database instances. This is why fine-grained host-based analysis as in Galapagos is particularly suitable for ITBVM, but the framework can take advantage of any available dependency discovery.

We can distinguish several types of dependencies, explicitly in an element dependency type or implicitly by source and target types. Different optimization tasks rely on different types of dependencies. E.g., if we plan to optimize CPU utilization we can restrict ourselves to dependencies that trigger CPU usage. This corresponds to considering control-flow dependencies only. Other IT management tasks, e.g., recovery planning, need other dependency types.

Some optimization work for homogeneous environments uses similar dependency graphs, for instance, the stream processing optimization system described in [25] or the optimization for a known large 3-tier application in [19]. For general services engagements, however, we cannot assume that such graphs have ever been completely documented, and certainly not in a consistent and up-to-date way. Furthermore, we are currently not aware of such work with multiple layers of object inclusion as we use it here.

3 A Concrete Sample Optimization Problem

In this section, we give a concrete sample of business-value driven IT optimization that illustrates the overall approach from Section 2. The sample model can be considered a simple showcase of optimizations that can be performed with information from a mostly blackbox system. In Section 3.1, we present a blackbox model where only the IT entry points and their values are known. In Section 3.2, we refine that model by discovered IT dependencies. In Section 3.3, we describe the concrete languages and tools used for implementing the model and solving it.

3.1 Blackbox Model

We consider a set $S$ of servers and a set $E$ of IT entry points. We assume that we can obtain the average utilization $util_s$ for each server $s \in S$ for reasonably short intervals, say, once every few minutes. This is possible because in many services engagements one can find standard server instrumentation at the IT level, e.g., SNMP. The customer may already obtain the measurements with some IT management system or allow the services provider to directly
pull such measurements because typically the risk of revealing sensitive customer data through such measurements is low. Although we consider CPU time the sole bottleneck in the following example, our overall approach can handle multiple types of resource constraints. We also assume we can observe the current input and output rates \( \text{in}_e \) and \( \text{out}_e \) for each entry point \( e \in E \). Due to the assumed blackbox nature of the services engagement, we cannot assume direct measurements of traffic rates on inner connections, nor advance knowledge of what entry point causes utilization on which servers.

As to control parameters for the optimization, only admission control is possible in this scenario; too little is known about the rest of the system to perform resource assignments. In other words, we are mainly interested in optimizing the actual input rate \( \text{in}_e \) to admit for each entry point out of available input \( \text{in}^\text{max}_e \).

A value \( \text{val}_e \) is given for each entry point, which indicates the average value per output at that entry point. This value can change between optimization periods based on the current business situation or a new measurement of perceived value. The objective function to be maximized is

\[
z = \sum_{e \in E} \text{val}_e \cdot \text{out}_e.
\]

In this example, our main constraints deal with CPU utilization per server. We define a constant \( \text{util}^\text{max} \) for a desired maximum CPU utilization in order to avoid thrashing effects. For each server, the cumulative utilization \( \text{util}_s \) is the sum of unknown utilizations \( \text{util}_{s,e} \) resulting from each entry point \( e \). Thus, we can add the following constraints:

\[
\forall s \in S, \quad \text{util}_s = \sum_{e \in E} \text{in}_e \cdot \text{util}_{s,e}. \quad (2)
\]

\[
\forall s \in S, \quad 0 \leq \text{util}_s \leq \text{util}^\text{max}. \quad (3)
\]

Furthermore, we assume that under these CPU utilization constraints, the computation yields one output per input (like in transactional requests) and is both loss-free and fast enough so that the outputs in one optimization period correspond to the inputs of the same period:

\[
\forall e \in E, \quad \text{out}_e = \text{in}_e. \quad (4)
\]

Finally, we cannot admit more input than we get. Furthermore, we may want to admit at least a small rate \( \text{in}^\text{min}_e \) for each entry point (as long as we get so much input), so that we can keep the value analysis for this entry point up to date. That observation results in the following constraint:

\[
\forall e \in E, \quad \min(\text{in}^\text{min}_e, \text{in}^\text{max}_e) \leq \text{in}_e \leq \text{in}^\text{max}_e. \quad (5)
\]

Solving to maximize the objective function \( z \) gives optimal values for all variables \( \text{in}_e \). These values are fed to the control subsystem, which reconfigures the admission control to ensure that these input rates take effect for the next epoch of system operation.

### 3.2 Using Dependency Information

If we are not completely restricted to a black-box model, but have the opportunity to discover dependencies, we can improve on the model above. As described in [8], it is not trivial for a services provider to obtain discovery authorizations and IT-level credentials, but we carefully crafted Galapagos to strongly improve the likelihood and speed of passing approval processes.

We transform the discovered dependencies as additional constraints in the optimization model as follows. Let \( G \) be the set of IT objects and \( T \) the set of dependency types. The dependencies are a set \( D \) of triples \((g, h, t)\) with \( g, h \in G \) and \( t \in T \). Let \( I, O \subseteq G \) be the input and output elements, respectively, e.g., URLs. We first need to map these elements to the entry points given from the business level, say \( I_e \) and \( O_e \) for each entry point \( e \in E \). For URLs (such as \text{http://cloudmagicl3.wf.ibm.com/BVM/*}), this mapping can be automated: the wildcard expressions used to describe entry points and those configured on the entry web servers need to be broken into common subsets.

Then, we proceed as follows:

1. We restrict the dependencies on the software object layer to those that imply control flow and thus CPU utilization dependency. This is a subset \( T' \) of the type set \( T \). If the model or the concrete discovery tool used is not so detailed, we retain all dependencies.

2. For each entry point \( e \), we derive the set \( G_e \) of all objects that depend on an input object from this entry point, i.e., all successors at any distance from \( I_e \) where only dependencies of types in \( T' \) count. Here, we also take into account the fact that in real middleware systems, dependencies can be configured at different levels in the object hierarchy as shown in Figure 2. Consequently, if the discovery tool yields the information that an IT object \( A \) depends on IT object \( B \), then the implication is that \( A \) also depends on \( B \)’s parents and children in the object hierarchy.

3. Now, we set \( \text{util}_{s,e} = 0 \) for all servers \( s \not\in G_e \). Here, we assumed that the dependencies of a software object on the underlying server are part of \( D \) and that the types of such dependencies are in \( T' \).

4. If \( s \not\in G_e \) for all \( e \in E \), we can delete server \( s \) from the set \( S \) for the given optimization problem.

In the following section, we will see that this can yield a significant simplification of the model, which results in improved model accuracy and faster solving times.
3.3 Implementation

We implemented the optimization in AMPL (A Mathematical Programming Language) [3], a standard optimization language supported by a number of solving tools. We used the open-source solver GLPK (GNU Linear Programming Kit) [13]. Beyond the current sample, our goal is to define a specialization of AMPL to data-center concepts that can be used generally in business-value-driven IT optimization. The evaluation of the dependency information is done in SQL over the result database of the Galapagos tool.

4 Evaluation

To evaluate the practicality of the proposed approach we performed experiments on two setups: a controlled scenario consisting of a small three-tier architecture, and a real enterprise data center where we performed the discovery, but only simulated the optimization. In both cases, we assume very little other instrumentation, corresponding to the overall customer assumptions described in Section 2.

4.1 Controlled Scenario

The architecture and component connectivity of our test setup called LAB is shown in Figure 3. For space reasons, we do not show individual applications and databases in this figure, although Galapagos can generate figures at that level of detail. In the corresponding scenario, an enterprise application implements three business functions, each corresponding to one IT entry point (recall Section 2.2). Execution of the different business functions consumes different amounts of resources. The optimization framework does not know these amounts a priori. The enterprise application deployment spans two web servers (located at hosts cloudmagic13.wf.ibm.com and cloudmagic14.wf.ibm.com) and two application servers (located at hosts cloudmagic24.wf.ibm.com and cloudmagic29.wf.ibm.com). Incoming HTTP requests specify the desired business function by a URL. Following execution, the output is persisted in a database (located at cloudmagic27.wf.ibm.com) along with meta-data such as the name of the business function, the time it took to process the request, and the requester’s IP address.

We implemented a web application (distinct from the enterprise application) through which users can view the entries in the database. In the LAB scenario, we use the popularity of the outputs of the different business functions as a simple example of measuring perceived value. For instance, if the outputs of one business function have been selected 100 times by users, and the outputs of another function only 10 times, we consider the value of the first outputs 10 times higher than the value of the second business function.

In the customer situation for which this scenario is a small sample, we are primarily concerned with CPU utilization, and there are more potential inputs than can be handled with the allotted resources. Hence admission control for maximizing the value of the results is our goal: “Given the current perceived values of different outputs and the observed overall resource utilization in processing, how do we maximize the value?”

Our management system consists of the instrumentation for per-server utilization measurements and a control point on each web server, i.e., at the input entry points. We used Apache JMeter [17] for generating the incoming traffic for the enterprise application. We measure the CPU utilization at each host and the input request rate at each entry point. These measurements correspond to the input parameters for the model described in Section 3.1. We first use a sequence of data sets of measured per-server utilization $util_e$ together with measured input rates from the same periods to calibrate the utilization parameters $util_e, e$ per server and entry point. Then, within shorter periods, we optimize for the current perceived values $val_e$ and the current input rates $in^e_{max}$ using the GLPK solver.

The control point implementation leverages the mod_gos module of the Apache web server, which is the standard module for configuring admission control. The optimal input rate for each entry point is computed by the solver. These values are pushed into a local management component for the web server, which translates them into concrete web server settings.

4.2 Real Enterprise Scenario

Next, we analyzed a real enterprise environment consisting of 536 servers. We performed discovery using Gala-
The discovery showed that among 536 servers, 99 were web servers accessed for business functions, 132 WebSphere application servers, and 332 database servers. We recognize 2,414 input URLs on the web servers, i.e., independently configured URLs that might be handled by different application modules. These URLs may have wildcards. We consider these as the entry points. The overall statistics about both discovered environments is presented in Table 1.

We have also discovered detailed information about server-to-server, URL-to-application, and application-to-database dependencies. A picture showing these dependencies similar to Figure 3 or even Figure 2 would not fit in this paper. Instead, in Figure 4 we provide a histogram showing the distribution of the number of servers directly serving requests coming to each of the 99 web servers. Not surprisingly, most URLs required processing by three servers that correspond to the common three-tier architecture. Interestingly, three input servers were connected to a big application formed by 33 servers.

4.3 Measurements

We measured the performance optimization times for both scenarios using the GLPK solver for the business-value based optimization model from Section 3. We compared the pure black-box model from Section 3.1 with the model enhanced by dependency information described in Section 3.2.

The solver ran on a server with two Intel Xeon 2.8 GHz, 512 KB cache CPUs and 4 GB of RAM. The solver server was connected to a database server that contained performance data, discovered connectivity data, and the values of the different URLs (entry points). We only measured CPU times of the actual optimization solving times, corresponding to two assumptions: Obtaining updated data at least for input rates will not be the bottleneck, and calibration of the parameters $util_{a,e}$ from measured data points can take place much less frequently than the main real-time optimization.

As we got the discovery results from the ENTERPRISE scenario from a different engagement, we had to randomly generate the utilization and value data for it based on typical values we observed.

We used the Auto-pilot benchmarking suite [26] to minimize the effects of external factors on our benchmarks. We ran each experiment at least ten times and used Student-\(t\) distribution to calculate 95% confidence intervals. We ensured that half-widths of the confidence intervals were less than 5% of the mean for each measured time component. We measured the user-level time necessary to perform the actual solving operation and system time that mostly corresponded to fetching input arrays of data from the system cache. We prefetched the data from the disk before the experiments as this better corresponds to the expected real-life situation.

Figure 5 shows the solver execution times for our LAB (left) and ENTERPRISE (right) experimental setups. Values in the left part are in milliseconds, in the right part in seconds.
lem reduced the problem size and significantly reduced the solving time. Already for our small LAB setup the optimization decreased processing times by 21%, and for the ENTERPRISE setup the time decreased by a factor of 2.9. This is because the solving time has a non-linear dependency on the problem size.

Note that while our demonstration is on a rather simple model, we expect the additional dependency information to become more and more useful when the performance models get more complex, because the calibration times and solving times will rise, and typically much more than linearly, with the number of parameters. One can speculate that discovery-based optimization can make it possible to perform real-time performance optimizations of environments consisting of thousands of servers, while without such optimization typical solving times will exceed minutes and make it impossible to operate in real-time mode.

A general caveat is that no discovery technique has perfect coverage, just like extrapolated statistics about past utilizations, input rates, and values never have perfect validity for the future. Still, as motivated in the introduction, not even complete and up-to-date dependency information is typically available for significant parts of an enterprise, and even less are perfect performance models. Static dependency discovery can play a significant role in getting an initial understanding and model of an enterprise, in particular in services engagements, which are typically of limited duration. If an optimization framework such as ITBVM remains in place, over time additional instrumentation, e.g., in the network, as well as cross-calibration with the utilization statistics, can be used to continuously improve the models.

5 Related Work

The distinguishing aspect of ITBVM is to link business value to IT for environments that are highly heterogeneous and come with little documentation. These are two standard characteristics of large IT environments that one encounters in real services engagements for IT transformation and optimization.

Linking business value to IT has recently been a major goal at the enterprise boardroom level [7, 21]. At this level, the focus lies on organizational governance structures, while we concentrate on concrete optimizations. There is also a large body of work in enterprise business value analysis and optimization from a variety of view points, e.g., financial accounting, pricing strategies, and supply-chain analysis. However, one has only recently begun to envision applying such analysis techniques to large-scale IT optimization, mainly because information about the business use and dependencies in large-scale IT environments was either incomplete or entirely missing.

IT optimization for small environments of known structure is often based solely on job priorities or policies, but in this area, real optimization based on values exists. The values usually come in the form of SLA requirements, which one may assume to have been based on business values. Examples include job scheduling algorithms, see [1] for an overview of types, optimization for a large but known 3-tier architecture [19], optimization for a Services Oriented Architecture (SOA) among providers of services with the same functional interface, and similar work where services adhere to other interfaces such as grid, autonomic, or cloud computing [20, 14, 22, 15, 11, 10, 23, 24]. ITBVM can be applied to use information available from, e.g., BPEL-designed environments. Unfortunately, it is still very uncommon to see enterprises composed purely or at least mostly with any of these mechanisms. Therefore, discovery-based connectivity of business-level models and IT is still the only viable solution for the vast majority of the real enterprises.

Blackbox analysis in the sense of primarily using performance data with only few structural data also exist in lower-level areas of IT optimization such as OS profiling [9].

Discovery of IT assets can be performed by scanning the network, listening for the network traffic, and logging in to individual servers, see, e.g., [5, 2, 6, 18, 12]. Within ITBVM all these approaches can be used, and indeed, as no discovery method is 100% reliable, multiple methods should be used whenever possible. However, this may not always be possible in a services engagement. In [8], we describe challenges we encountered and how we overcame them for Galapagos. A specific advantage of using static configuration information as we did in the concrete scenarios in this paper is that it enables the discovery of finer-grained dependencies as shown in Figure 2.

6 Conclusions

We have designed a framework ITBVM for expressing business-level values, mapping them to IT entry points, following IT dependencies from these entry points, and subsequently performing global optimizations of IT resources for the best business results of IT operation. We have focused on the common case in services engagements, where initially there is no complete, up-to-date, or consistent information about IT applications and data and their dependencies at the scale of the desired IT optimization. Furthermore, one can typically assume very little common instrumentation to obtain consistent performance data on an ongoing basis. We therefore started with a black-box performance model and showed how even in that case, business-value based optimization is possible. We subsequently showed how dependency discovery, in particular of a class that can be rapidly deployed by a services provider, can improve the accuracy of the models and the efficiency of
solving the related optimization problems. We have demonstrated this with a controlled lab scenario as well as an enterprise scenario, using an open-source solver.

Acknowledgements We thank Alain Biem, Manoj Kumar, Kostas Magoutis, Vijay Naik, H. J. Siegel, and Włodzimir Zadrozny for interesting discussions. Part of this work was funded by US Government contract H98230-05-G-0191.

References


