RC24048 (W0609-014) September 7, 2006
Computer Science

IBM Research Report

Predicting Labor Cost through IT Management Complexity Metrics

Yixin Diao, Alexander Keller, Sujay Parekh
IBM Research Division
Thomas J. Watson Research Center
P.O. Box 704
Yorktown Heights, NY 10598

Vladislav V. Marinov
Department of Computer Science
International University Bremen
Campus Ring 1
28759 Bremen
Germany

Predicting Labor Cost through IT Management Complexity Metrics

Yixin Diao, Alexander Keller, Sujay Parekh
IBM Thomas J. Watson Research Center
P.O. Box 704
Yorktown Heights, NY 10598, USA
Email: {diao|alexk|sujay}@us.ibm.com

Vladislav V. Marinov
Department of Computer Science
International University Bremen
Campus Ring 1, 28759 Bremen, Germany
Email: v.marinov@iu-bremen.de

Abstract— **We propose a model for relating IT management complexity metrics to key business-level performance metrics like time and labor cost. In particular, we address the problem of quantifying and predicting the value that automation and IT service management process transformation will yield before their actual deployment. Our approach looks at this problem from a different, new perspective by regarding complexity as a surrogate for potential labor cost and human-error-induced problems: It consists in (1) assessing and evaluating the complexity of IT management processes and procedures, (2) separately measuring business-level performance metrics (3) relating the collected complexity metrics to business-level performance metrics by means of a quantitative model, and (4) validating the model through a field study. Algorithms are presented for selecting a subset of the complexity metrics to use as explanatory variables in a quantitative model and for constructing the quantitative model itself. Besides improving decision making for deploying automation technologies and transforming IT service management processes, our quantitative IT management complexity model can help service providers and outsourcers predict the amount of human effort and skills that will be needed to provide a given service, thus allowing them to more effectively evaluate costs and benefits of automation technologies and IT management process transformations.**

I. Introduction

One of the key promises of IT to business is to deliver greater efficiencies through faster processing, labor savings and by enabling novel capabilities. In particular, IT departments of enterprises and service providers are increasingly looking to automation and IT process transformation as a means of containing and even reducing the labor costs of IT service management. However, there is much anecdotal evidence as well of a *reduction* in efficiencies and productivity when new technology or transformed processes are introduced into business operations. This raises a critical question from the point of view of both the business owners and CIOs as well as service and technology providers: how can one quantify, measure and (ultimately) predict whether and how the introduction of a given technology can deliver promised efficiencies?

The two most popular strategies that aim at reducing labor cost in IT service management – the factor dominating total cost of ownership – are (1) the deployment of automation to reduce human involvement, and (2) the transformation of IT service management processes so that they comply with standardized frameworks, such as the IT Infrastructure Library (ITIL) [1], or the enhanced Telecom Operations Map (eTOM) by the TeleManagement Forum. Both strategies have in common that they focus on the creation of standardized, reusable components that replace today's custom-built solutions and processes — the former at a system level, the latter at a process level. While both service providers and IT management software vendors heavily rely on such qualitative statements to justify investment in new technologies, a qualitative analysis does not provide direct guidance in terms of *quantitative* business-level performance metrics, such as labor cost, time, productivity and quality.

In previous work [2], we have introduced a framework for measuring IT system management complexity, which has been subsequently extended to address the specifics of IT service management processes [3]. The framework relies on categorical classification of individual complexities. For example, for characterizing the execution complexity of an individual task, three categories are defined based on the degree of automation: automatic, tool-assisted and manual. A complexity score derived from such categorization does not directly reveal the actual labor reduction or cost savings.

In order to address this issue, this paper describes an approach to relate the metrics of our IT management complexity framework to key business-level performance metrics. Our proposal is to develop a quantitative model based on a qualitative description of IT complexity parameters together with quantitative calibration data of the IT process. We believe such a *hybrid* approach is more practical than a pure qualitative or quantitative approach, since extensive quantitative IT process measurements are notoriously difficult to obtain in the field.

Such a model has many benefits: First, the model can be used to assist in return-on-investment (ROI) determination, either a-priori or post-facto. Across multiple products (for example, IT offerings from different providers), a customer can use models involving each product for a comparison in terms of the impact on the bottom line. Conversely, IT sales personnel can make a substantiated, quantitative argument during a bidding process for a contract.

Second, using the cost prediction based on the model, the customer can further use the model output for budgeting purposes.

Third, a calibrated model for a particular process can reveal which are the important factors that contribute to the overall complexity of the process, along with a measure of their relative contributions. IT providers can use this data to improve their products and offerings, focusing on areas which yield the largest customer impact.

Fourth, a particular process can also be studied in terms of its sensitivity to skill levels of individual roles. Because labor cost of different skill levels varies, this sensitivity analysis can be used for hiring and employee scheduling purposes.

Finally, process transformation involves cost/benefit analysis. These decisions can be guided by the quantitative predictions from the model.

As a key step towards this vision, the goal of this paper is to address the problem of quantifying and predicting labor cost for IT service management by means of a quantitative model that relates IT management complexity metrics to business-level performance metrics. It is organized as follows: Section II overviews our model for IT management processes, discusses its challenges and outlines our approach to address them. Section III details our quantitative model for IT complexity metrics and the calibration algorithms. In section IV, we perform a quantitative complexity analysis by applying our model to a small-scale user study. Section V reviews related work. Our conclusions are contained in section VI.

II. Quantitative Complexity Modeling

A. IT Management Complexity Metrics

Our discussion is based on the IT management complexity framework first described in [2] and further extended by [3]. Within the scope of this paper, we rely on a subset of the former model whose intent is to capture and quantify the complexity of a straight-line flow through a configuration procedure. Three different complexity dimensions – each having several complexity metrics – were identified, which are briefly summarized below in order to provide some background for the following discussion: *Execution Complexity* refers to the complexity involved in performing the configuration actions that make up the configuration procedure, typically characterized by the number of actions and the context switches[1] between actions. *Parameter Complexity* is the complexity involved in providing configuration data to the computer system during a configuration procedure. For examples, each parameter is assigned a *source score* on a linear scale from 0 – 6 based on how its value was obtained; low scores represent easily-obtained values (such as those surfaced earlier in the procedure) whereas high scores represent more obscure values (such as those that must be extrapolated from the system environment – a JVM path, for instance – or chosen based on experience). Finally, *Memory Complexity* takes into account the number of parameters that must be remembered, the length of time they must be retained in memory, and how many intervening items were stored in memory between uses of a remembered parameter.

[1]A context switch occurs between any two consecutive configuration actions that act upon different containers, such as the command-line interface of an operating system or an installation wizard for a software product.

B. Challenges

While this simple complexity model permits – in addition to capturing the complexity metrics – the direct recording of execution times, the latter is often difficult to obtain in practice. The complexity metrics are easier to describe/obtain in *qualitative* terms, since they do not require detailed user studies and can sometimes even be done by interviewing a subject matter expert. For example, in the case of parameter source scores (see above), it is relatively easy to describe whether a parameter is obtained from documentation or must be specified based on environmental constraints. Such a description is rather general and applies to many (or all) instantiations of the process, compared to quantitative data about, say, the time it takes to first come up with a value for the parameter, or some other encoding of its information content. Unfortunately, a complexity evaluation based on qualitative data alone does not provide direct guidance in terms of quantitative business-level performance metrics.

There are some challenges in taking this approach as many of the complexity metrics have categorical descriptions: First, the categories intuitively represent different degrees of information content or difficulty in obtaining the values. However, we need to assess whether the complexity metrics contained in the model are actually the right ones, i.e., whether they correlate with independently captured business-level performance metrics.

Second, as a starting point, we have chosen that metric values are expressed in a linear scale throughout the model. For example, by assigning a *freeChoice* (numerical value: 0) for the parameter source score, one indicates that this configuration parameter is clearly easier to specify than an *environmentFixed* parameter (value: 5). Although one may order these classifications according to their relative difficulty, the exact quantification of the increase in terms of complexity may vary. In particular, it is not clear if the linear scale for the parameter source score is appropriate, or if a polynomial (or even exponential) scale is more suitable.

Third, there is a need to capture the degree of expertise and the skill set of the *role* that is carrying out a task. This may impact the exact translation from qualitative value to quantitative value.

Building a quantitative model helps in calibrating the IT management complexity metrics by determining how each of the individual metrics correlates with the independently observed business-level metrics. By helping eliminate metrics that are not relevant, the quantitative model provides guidance on how to simplify the model while maintaining its applicability.

C. A 4-Step Approach to Predicting Labor Cost

Our approach to predicting labor cost through IT management complexity metrics is depicted in Figure 1. It consists of four distinct steps, which can be summarized as follows:

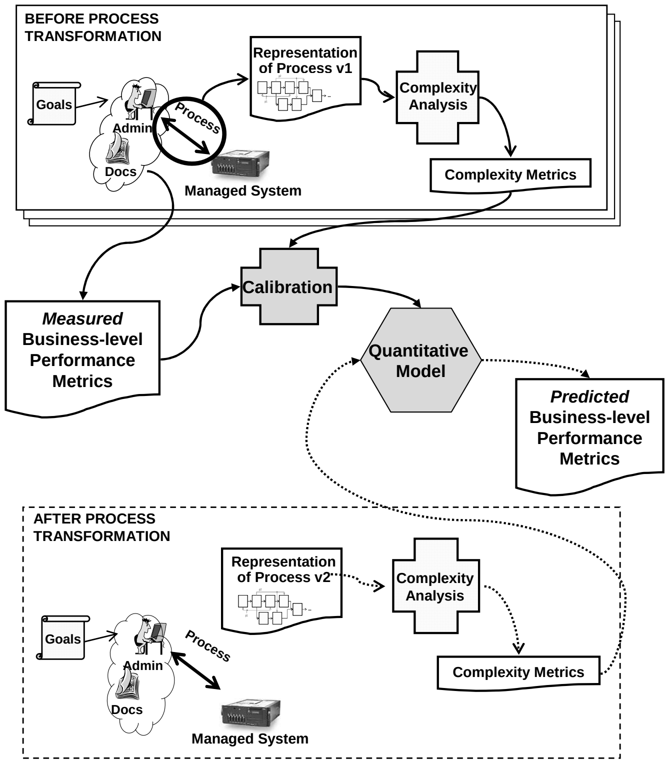
BEFORE PROCESS TRANSFORMATION
Goals
Admin
Docs
Process
Managed System
Representation of Process v1
Complexity Analysis
Complexity Metrics
Calibration
Measured Business-level Performance Metrics
Quantitative Model
Predicted Business-level Performance Metrics
AFTER PROCESS TRANSFORMATION
Representation of Process v2
Complexity Analysis
Complexity Metrics
Goals
Admin
Docs
Process
Managed System

Fig. 1. Approach to constructing and applying a quantitative model

1) Collecting Complexity Metrics: As depicted in the upper part of Figure 1, an administrator configures one or more managed systems, according to the goals he wants to accomplish. To do so, he follows a process that may either be described in authoritative documents, or as a workflow in an IT process modeling tool.

This process (called 'Process version 1' in the Figure) reflects the current ('as-is') state of the process, which serves as a baseline for our measurements. While carrying out the configuration process, an administrator logs – by interacting with a web-based graphical user interface [3] – the various steps, context switches, and parameters that need to be input or produced by the process. In addition, the administrator assesses the complexity for each of the parameters, as perceived by him. The complexity data that an administrator records by means of the web-based graphical user interface is recorded by our tooling on a step-by-step basis. The key concept of the process representation is that it is 'action-centric,' i.e., the representation decomposes the overall process into individual, atomic actions that represent the configurations steps an administrator goes through.

Once the process has been carried out, the process capture is input to the Complexity Analysis Tool we implemented, which contains our scoring algorithms for execution, parameter, and memory complexity. The tool also outputs aggregated complexity metrics, such as the maximum number of parameters that need to be kept in memory during the whole procedure, the overall number of actions in the process, or the context switches an administrator needs to perform. Both the individual and the aggregated complexity metrics will be used to construct the quantitative model.

We note that by assigning a complexity score to each configuration action, it is possible to identify those actions in the process (or combinations of actions) that have the greatest contribution to complexity (i.e., they are complexity hotspots). As a result, the methodology facilitates the task of process designers to identify targets for process transformation and optimization: If, for example, a given step has a very high complexity relative to other steps in the process, a designer obtains valuable guidance on which action(s) his improvement efforts need to focus first in order to be most effective.

2) Measuring Business-level Performance Metrics: The second step consists in measuring the business-level performance metrics. The most common example of a business-level metric, which we use for our analysis, is the time it takes to complete each action in the process, and the aggregated, overall process execution time. This is also commonly referred to as the *Full Time Equivalent (FTE)*. Labor Cost, another key business-level performance metric, is derived in a straightforward way by multiplying the measured FTEs with the administrator's billable hourly rate.

Note that while our tooling allows an administrator to record the time while he captures the actions in a configuration process, the time measurements are kept separately from the complexity metrics.

3) Model Construction through Calibration: The step of constructing a quantitative model by means of calibration, depicted in the middle of Figure 1, is at the heart of this paper and is explained in detail in the following section III. Calibration relates the complexity metrics we computed in step 1 to the FTE measurements we performed in step 2. To do so, we have adapted techniques we initially developed in [4] (where we were able to predict the user-perceived response time of a web-based Internet storefront by examining the performance metrics obtained from the storefront's database server) to the problem domain of IT management complexity. The purpose of calibration is to set the time it takes to execute a configuration process in relation to the recorded complexity metrics, i.e., the former is being explained by the latter.

4) Predict Business-level Performance Metrics: Once a quantitative model has been built based on the 'as-is' state of a process, it can be used to predict the FTEs and therefore the labor cost for an improved process that has been obtained through process transformation based on analyzing and mitigating the complexity hotspots that were identified in step 1. This 'to-be' process is referred to as 'Process version 2' in the bottom part of the Figure as it accomplishes the very same goal(s) as the 'as-is' version 1 of the process. It is therefore possible to not only apply the quantitative model that has been developed for the 'as-is' state in order to estimate the FTEs from the complexity metrics of the 'to-be' state, but also to directly compare both the complexity metrics and the FTEs of the before/after transformation versions of the process. The difference between the 'as-is' and the 'to-be'

FTEs and, thus, labor cost yields the savings that are obtained by process transformation.

For the scope of this paper, we use the measured FTEs that were collected by running a similar, but slightly different process (namely, the installation of a different product within the same family) to validate our quantitative model that is build by calibration. However, we envision a future scenario where our validated quantitative model is built directly into an IT process modeling tool, so that a process designer can simulate the execution of an IT management process during the design phase. The designer is then able to obtain the predicted FTEs by running the complexity metrics that were automatically collected from the process model – by means of a plugin – through the quantitative model. (See [3] in which we detail extensions of our complexity model to address the specifics of IT process models.) It is then possible to determine if the process transformation is likely to yield the expected savings in terms of labor cost/FTEs, or if further optimization is needed before the transformed process is ready for deployment.

III. A Quantitative Model

This section describes the method of building a quantitative model to predict labor cost based on complexity metrics. The model structure and usage scenarios are first discussed in detail. Subsequently, we explain the algorithms for model construction and exploitation.

A. Model Structure and Usage Scenarios

Many system modeling methods have been studied and applied to solve real world problems in function approximation and system identification (e.g., [5], [6], [7]). However, most of them are used to model the input-output relationships of quantitative variables (e.g., arrival rate, response time).

In this paper, we need to build the model for IT management complexity metrics which – as mentioned in section I – have a strong qualitative nature and are represented with categorical descriptions. We choose to use the multiple linear regression method [8]. This is because the linear models tend to be more robust than nonlinear models such as neural network models, especially when a large number of model inputs are involved. The general form of our linear quantitative model can be written as

$$y = b_0 + b_1 x_1 + b_2 x_2 + \cdots + b_n x_n \tag{1}$$

where the explanatory variables x_i are used to represent the IT management complexity metrics (cf. section II-A), and the response variable y refers to the business-level performance metrics such as labor time. Their relationships are characterized through model parameters b_i. When multiple business-level performance metrics exist, different quantitative models can be constructed, each of which has a different set of model parameters.

The main scenario of using the quantitative model is to predict the labor cost (or FTEs/time) for the IT management processes. This is useful to validate the design of new processes when the time measurements can only be collected for the existing or related 'as-is' processes, but not for the new 'to-be' process. Although it is always preferred to capture and quantify how an administrator experiences the system in terms of both labor time and complexity, having a comprehensive user based study is typically difficult. The quantitative complexity model provides an alternative way to evaluate new products or processes with respect to their values in reducing the operational complexity.

B. Model Algorithm

The quantitative model is constructed and exploited as follows:

1) Baseline Evaluation: The baseline ('as-is') process is executed through a group of system administrators, and the time measurements for each operational task are recorded. In addition, the complexity of the process is evaluated through the IT management complexity model defined in [2] and [3].

2) Quantitative Model Calibration: The quantitative model is calibrated using the least square approach [5]. Least square operates by minimizing the sum of the squared deviations between the measured response data and the estimated values from the model. That is,

$$\min \sum_{k=1}^{K} \left(y(k) - \hat{y}(k)\right)^2 \tag{2}$$

where K is the total number of measurements, $y(k)$ is the k-th measurement, and $\hat{y}(k)$ is the k-th estimated value. Assuming the data follows a normal distribution, least square gives an unbiased estimation of model parameters b_i.

In order to have enough data to build an accurate model, the more explanatory variables we have, the more time measurements we need. Since the number of available time measurements is generally limited, we need to identify a small set of IT complexity metrics to be used as explanatory variables.

Moreover, having a small set of explanatory variables avoids "overfitting" the modeling data, thereby preserving the adaptability to new data. The quantitative model is built based on the data from an existing process in order to characterize the relationship between the complexity metrics and the business-level performance metrics. Although more explanatory variables provides better modeling capabilities and reduces modeling errors for this data set, this does not necessarily increase the quality of the model. The extra variables tend to model the "noise" instead of the true relationship; which will impair its prediction capability for a new 'to-be' process. To solve this problem, we enhance the metric selection process with a cross validation technique [9]. It divides the available modeling data set to several subsets, and only uses a part of them as the training data to build the model. The remaining data sets are used as testing data to assess the quality of the model and help determine if we need to include more complexity metrics into the explanatory variable set. The above process is repeated in order to use different data subsets for training and testing, so that all the available data can be effectively used.

We identify this set of dominant metrics using an incremental approach: It incrementally selects the best metric based on what the current model does not explain. The steps for metric selection are summarized as follows.

1) Initialization. Define an initial model with zero explanatory variables.
2) Evaluation. Given the explanatory variable set, use the least square method to compute the model parameters and the modeling error.
3) Cross correlation. Compute the correlation coefficients between the modeling error and the complexity metrics that are not in the set of explanatory variables.
4) Model increment. Add the complexity metric that gives the largest correlation coefficient into the explanatory variable set.
5) Test for termination. If the model quality stops improving, then metric selection is complete; otherwise, go to Step 2.

More details on the algorithm can be found in [4]. We will also illustrate the metric selection process in Section IV.

Identifying the dominant complexity metrics also helps further analysis as we can focus our attention on this small set of complexity indicators. For example, we can assess the quality of the studied product or process by evaluating the average and standard deviation of the dominant complexity metrics identified above. Furthermore, we can estimate what would be the loss if certain metrics were not available, and what the redundant metrics are. Collecting redundant metrics can be costly, whereas their incremental benefit is limited. On the flip side, eliminating redundant metrics helps simplify our IT management complexity model and thereby improves its consumability.

3) Model Accuracy Assessment: The accuracy of the quantitative model constructed above is quantified through the R-Square metric (R^2) and the Root Mean Squared Error ($RMSE$). The R-Square metric is defined as

$$R^2 = 1 - \frac{var(y - \hat{y})}{var(y)} \tag{3}$$

where $var(.)$ denotes the variance. The R^2 metric quantifies the amount of variation for a response variable that can be explained by the model. A R^2 value of 1 indicates all the variability has been captured by the model; $R^2 = 0$ means fitting the data into a constant where the response data variability is not captured at all. Note that it is also possible to get a negative R^2 value if the model performs worse than just fitting to a constant. The Root Mean Squared Error is defined as

$$RMSE = \sqrt{\frac{1}{K}\sum_{k=1}^{K}\left(y(k) - \hat{y}(k)\right)^2} \tag{4}$$

A high quality model with a large R^2 value and a small $RMSE$ value indicates a consistent mapping between the complexity metrics and the labor time, that is, both of them accurately represent the studied process. On the other hand, a low quality model may indicate problems in biased complexity metrics due to improper complexity evaluation, or skewed labor time due to an incomplete user study.

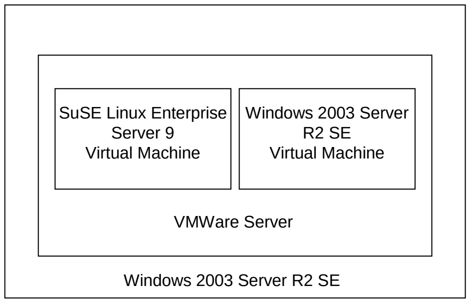
SuSE Linux Enterprise Server 9 Virtual Machine
Windows 2003 Server R2 SE Virtual Machine
VMWare Server
Windows 2003 Server R2 SE

Fig. 2. System Architecture

4) Model Extrapolation: The constructed quantitative model is applied to a transformed ('to-be') IT management process to predict the labor cost. As shown in Figure 1, the model uses the complexity metrics as inputs and predicts the business-level performance metrics for that process.

IV. Evaluation

A. System Environment

In order to construct and validate our quantitative model for predicting labor cost through IT management complexity metrics we have performed a small user study to collect time data from several IBM Tivoli product installations. The system environment throughout the experiments consists of six physical machines connected to the IBM internal network, each hosting one or more virtual machines.

All physical machines were running Windows 2003 Server R2 Standard edition as an operating system and VMware Server 1.0.0 was used as a platform to host the virtual machines. An illustration of the system architecture used during the experiments can be seen in Figure 2.

The Windows ***sysprep*** tool and the VMWare Console were used to prepare Windows 2003 Server R2 Standard Edition and SuSE Linux Enterprise Server 9 installation images. Every participant in the user study received a virtual machine with the required operating system image before each installation. Virtualization using pre-built images ensured that all experiments were performed in an identical operating system environment and allowed us to compare and analyze the data collected from the different participants. Furthermore, virtualization also improved the time efficiency of the user study as the time required for installation of an operating system and preparing it for an experiment was reduced to approximately five minutes by simply cloning a virtual machine and starting it.

B. Data Capture Methodology

Throughout the user study each participant was asked to install five IBM Tivoli products:

- IBM Tivoli Composite Application Manager (ITCAM)
- IBM Tivoli Identity Manager (ITIM)
- IBM Tivoli Provisioning Manager (TPM)

- IBM Tivoli Monitoring (ITM)
- Configuration Discovery and Tracking (CDT) feature for IBM Tivoli Change and Configuration Management Database (TCCMDB)

All of the above are implemented as J2EE applications whose setup required installation and configuration of a web application server, database server, directory server and some additional prerequisite software. Some of the installation images provided an automated installer that bundles the application, its underlying middleware and any applicable fixes; in other cases, the installations had to be performed completely manually. The participants evaluated each installation step by taking a screen capture and annotating the current context as well as the configuration parameters that were produced or consumed, according to the complexity model described in [2]. Afterwards, the collected data were run through the Complexity Analysis Tool which generated the complexity metrics for each installation step as well as for the whole process. Furthermore, every participant recorded the time spent for each step.

For the subsequent data analysis, we only considered the busy time, i.e., the time spent reading instructions, determining the right actions and parameter values, entering parameters into the install wizard, and taking notes for a specific step. The time after a participant had completed a screen of data capture and waited for the next data screen to be displayed was considered idle time and discarded.

The five Tivoli products used in the field study (each product being installed by several different participants, leading to a total of twelve independent installs) allowed time data to be correlated with various complexity metrics and thus a comprehensive quantitative model to be built. For example, the participants were asked to perform a manual installation of ITIM in which case all prerequisite software had to be installed separately (including manually applying fixes to the application server). This process involved a higher number of context switches; the corresponding time data helped us analyze the amount of time spent on context switches. On the other hand, the installation of TPM featured a prerequisite installation wizard, where a higher number of produced parameters had to be stored in the participant's memory and reused later on during the installation process. By analyzing how much time it took to retrieve data produced earlier in the process we were able to associate memory complexity metrics with time. Last but not least, providing input data during the installation of TPM required an extensive study of the product documentation and the system environment. This led to a significant increase in the parameter complexity and required to correlate parameter source complexity with time.

C. Data Capture Example: 2-Node ITCAM Installation

In order to illustrate our experiments, we briefly describe what the participants were required to do for installing ITCAM for the IBM WebSphere Application Server (WAS). ITCAM for WAS provides problem determination, availability monitoring and performance analysis for enterprise WAS applications running on Windows, UNIX, OS/400, and z/OS environments. It consists of one Managing Server and one Data Collector (agent) on each of the managed resources. Each Data Collector monitors an application server and communicates essential operational data to the ITCAM Managing Server. The participants were asked to perform a 2-node topology installation by installing an ITCAM Managing Server on one machine and an ITCAM Data Collector on another machine. Furthermore, as a clean operating system build was used as a starting point, all prerequisite software had to be installed and configured. We outline the major steps that the participants had to perform in order to install the product as well as the data they had to record.

1) Create User on the ITCAM Server: As the first step in ITCAM Managing Server installation, an installation user had to be created in the operating system. Following the install manual, the participant configured the username with a parameter source complexity of **documentationDirect[2]** (the number in brackets represents the numerical value on a linear scale that corresponds to the textual categorization). In addition, a password was specified; since the password could be chosen freely, its parameter source complexity was **freeChoice[1]** (for further details regarding the IT complexity model the reader is referred to [2]).

2) Install Prerequisite Software: The next step was to install Microsoft Windows Services for UNIX (SFU). Invoking the SFU installer caused a context switch from the operating system to the SFU installer. The participants had to select custom installation from the set of possible choices as described in the installation guide and thus supplying another parameter that has source complexity of **documentationDirect[2]**. In addition, a selection of the SFU packages to be installed had to be made. This was considered as another parameter that had to be retrieved from the product documentation. At the end of the SFU installation the system PATH variable had to be updated by adding adding the locations of the SFU executables. This step required providing two environmentally fixed parameters of source complexity **environmentFixed[5]**. After the installation of SFU the system mode had to be changed through a shell command, thus performing a context switch to the operating system and providing a **documentationDirect[2]** parameter, specified in the installation guide.

3) Start ITCAM Installation Wizard: The ITCAM wizard had to be started, an activity which involved another context switch from the operating system to the ITCAM install wizard. The installation user and password had to be specified; these two parameters were produced earlier in the process and thus considered **internal[0]**. Then, the user was asked by the installation wizard to specify both the WAS installation image location as well as the WAS Fixpack image location. The image directories were at a specific location within the file system of the machine. Thus, this step involved providing two parameters that were **environmentFixed[5]**. The WAS installation and the application of the fixpack were carried out by the wizard without further user involvement. In the next step, another parameter of the same type had to be provided as

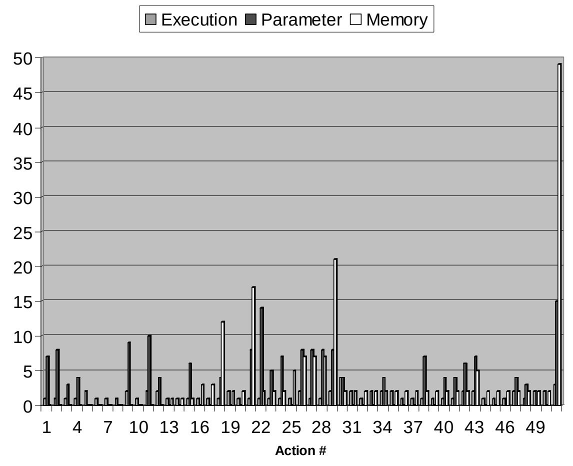

Fig. 3. Per-action complexity metrics from ITCAM install.

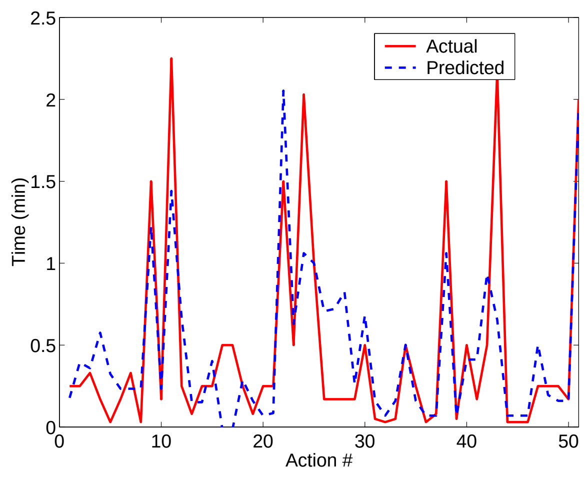

Fig. 4. Model calibration with data from ITCAM install.

the location of the DB2 installation image had to be specified. Then, the WAS server had to be restarted by the user by means of two activities that involved a context switch from the ITCAM installer to the operating system. No additional parameters were required from the user.

4) Install WebSphere Application Server on the Managed System: After completing the installation of the ITCAM Managing Server on one system, the participants had to perform a context switch to a second machine to install the Data Collector. This was started by installing an instance of the WAS server which mainly involved accepting a set of default settings. In addition, a context switch was invovled from the operating system to the WAS installation wizard.

5) Install ITCAM Data Collector on the Managed System: After completing WAS install, the participants need to switch to the context of the ITCAM Data Collector installation wizard. Apart from accepting default settings, they were required by the wizard to provide both the hostname and the home directory of the previously installed ITCAM Managing Server. Thus, two environmentally fixed parameters were supplied — the hostname being part of the DNS and the home directory being part of the filesystem.

D. Evaluation Results

In this section we demonstrate and evaluate our quantitative model for predicting labor cost through IT management complexity metrics. To illustrate the operation of the model, we start with two field study experiments; both of them were conducted by the same participant. The data from installing ITCAM were first used to calibrate the model. Afterwards, the model was extrapolated to predict the installation time for a different product – ITM. The model accuracy was validated against the actual time measurement. Finally, we apply the model to all the installation data that we have, and discuss the insights from these results.

1) Model Calibration: To calibrate the model, we need both the complexity metrics and time measurements. According to the complexity model defined in [2], we consider a total of ten complexity metrics which include two execution complexity metrics (Base Execution Complexity, Context Switch Distance), four parameter complexity metrics (Base Parameter Complexity, Parameter Source Complexity, Parameter Source Context Distance, Parameter Adaptation Indicator), and four memory complexity metrics (Memory Size, Memory Additions, Memory Latency, Memory Depth).

Figure 3 provides a per-action view for all 51 actions involved in the ITCAM installation. The x axis indicates the actions, and the y axis indicates the metric values. Although all ten complexity metrics can be plotted separately, we aggregate them into three high-level views that correspond to execution complexity, parameter complexity, and memory complexity (the three complexity dimensions defined in [2]).

Besides the complexity metrics, we also collected the per-action time measurements. They are shown as the solid line in Figure 4, where the x axis indicates the actions and the y axis indicates the labor time. Based on the complexity metrics and the time measurements, a quantitative model was constructed using the least squares approach. The predicted time is shown as the dashed line in Figure 4, which indicates a close fit between the measured data and the predicted values. The R^2 error is 0.61, which means the model can explain 61% of the variability in the time data. The root mean square error is 0.37; that is, the average difference between the measured and predicted time is 0.37 minutes.

Furthermore, we perform metric selection to determine the dominant complexity metrics. We start from an initial model and perform cross correlation between the ten complexity metrics and the modeling error (that is, the measured labor time, since the initial model with zero explanatory variables could not give any useful prediction). Figure 5 shows the absolute values of the correlation coefficients. The fourth complexity metric (i.e., Parameter Source Complexity) shows the highest value (0.68) and will be added into the explanatory variable set to build the model. The relatively large correlation

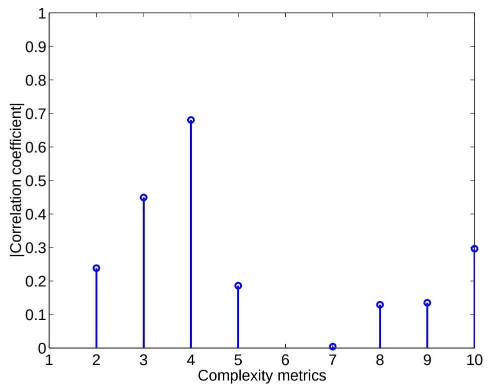

Fig. 5. Correlation analysis with data from ITCAM install.

TABLE I
Complexity metrics ordered according to their correlation to the labor time.

	Complexity Metric	R^2	b_i
1	Parameter Source Complexity	0.46	0.22
2	Memory Additions	0.51	0.31
3	Base Parameter Complexity	0.54	-0.31
4	Memory Size	0.57	-0.08
5	Parameter Source Context Distance	0.57	-0.16
6	Context Switch Distance	0.59	0.09
7	Memory Depth	0.61	0.02
8	Memory Latency	0.61	-0.06
9	Base Execution Complexity	0.61	0
10	Parameter Adaptation Indicator	0.61	0

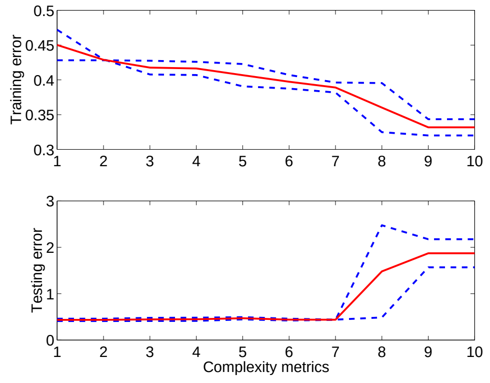

Fig. 6. Model cross validation with data from ITCAM install.

coefficient values also indicate a strong linear correlation between the complexity metrics and the labor time, so that building a linear model for labor time prediction is feasible.

Following the above metric selection process, the importance of each complexity metric can be determined and ordered as shown in Table I. Note that the metrics are ordered according to their contribution to the model, and thus follow a different sequence as the correlation coefficients in Figure 5. Although the metric with the highest correlation coefficient is the one that contributes the most to the model, the metric with the second highest correlation coefficient may not contribute much because it can be correlated to the first so that its contribution is discounted. This is the case for the third metric (Base Parameter Complexity) in Figure 5. Even if its correlation coefficient is second to that for the fourth metric (Parameter Source Complexity), its contribution to the model is not the second because Base Parameter Complexity is strongly correlated to Parameter Source Complexity.

Table I also gives the R^2 error for the model with increasing number of complexity metrics; it indicates that the quality of the model mainly is affected by the first few metrics. The last column of Table I gives the modeling parameters in Equation (1). The magnitude of the modeling parameter does not indicate its importance since the magnitude of the metric can be drastically different. In addition, the negative values of the modeling parameters indicate "overfitting" where the metrics are used to best fit into the data but do not improve the relationship characterization.

We use the cross validation technique to determine the right number of complexity metrics to be included in the explanatory variable set. Figure 6 displays how the root mean square error changes with respect to the number of complexity metrics used. The x axis indicates the complexity metrics in the same order as that in Table I, whereas the y axis indicates the modeling error. We separate the 51 actions from the ITCAM installation into two data sets, and iteratively use one for training and the other for testing. The top plot displays the modeling error during the training phase. The two dashed lines indicate the training errors for the two iterations; the solid line represents the average. Note that having more variables always improves the training quality. In contrast, as shown in the bottom plot, having more variables may impair the model quality due to overfitting. As a result, we only choose the first two complexity metrics, Parameter Source Complexity and Memory Additions, to build the model. The relative importance of these two metrics is consistent with the experience of system administrators as most of the labor time was spent in determining the right values for the required parameters. In addition, memory additions also result in longer labor time because the system administrators need to absorb the parameters produced by the system.

2) *Model Validation:* To validate the quality of the quantitative model constructed above, we consider a slightly different Tivoli installation process, namely the installation of the IBM Tivoli Monitoring (ITM) product. The quantitative model we built above is used to predict the ITM installation time based on the ITM complexity metrics. Then, we compare the predicted labor time with the measured time to evaluate how well the model performs in predicting the time.

Figure 7 shows both the actual labor time (solid line) and the predicted time (dashed line). The figure suggests a close fit, where the R^2 error is 0.60 and the RMSE is 1.31. The model is able to predict the labor time spike at action 4, which consumes five parameters that determine which product features to install. All of them have the Parameter Source Complexity

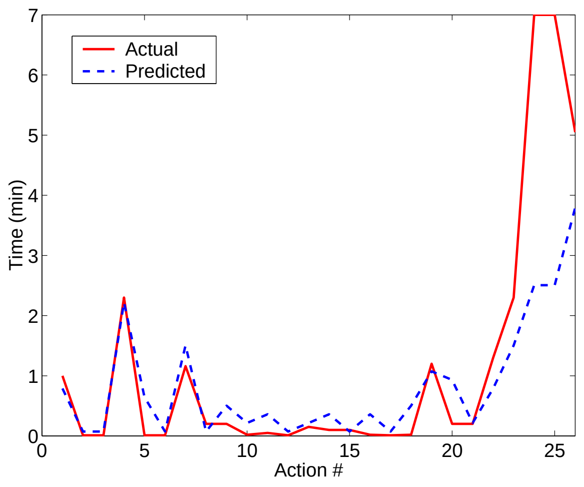

Fig. 7. Model validation with data from ITM install.

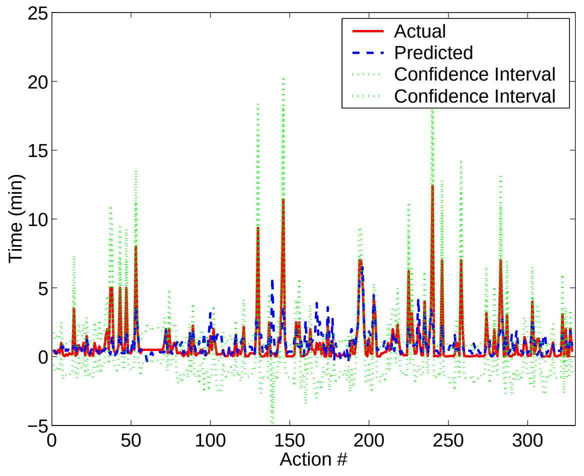

Fig. 8. Modeling results from all Tivoli installs.

as **documentationAdapted[3]**. However, the model does not provide an accurate prediction for actions 24-26. Based on the four to five parameters consumed by these actions, all with high Parameter Source Complexity (i.e., **bestPractice[4]**, **environmentFixed[5]**, **environmentConstrained[6]**), the model is able to predict a long labor time. However, the actual labor time is longer than expected. This may be due to the fact that a command line interface (instead of an installation wizard) is used, which complicates the task even further.

E. Accounting for Administrator Skills

Besides the ITCAM and ITM installations we considered above, we now apply the model to all the installation data that we have from the Tivoli install experiments.

Figure 8 gives an overview of the modeling results, where the solid line indicates the actual labor time, the dashed line indicates the predicted time, and the two dotted lines indicate the upper and lower bounds of the 60% confidence interval.

Out of all Tivoli install field study experiments, we have two distinct groups: Group 1 contains four Tivoli product installations (ITCAM, ITSM, ITM, and TPM), all of which were carried out by the same participant who is experienced in both systems management and complexity evaluation. Group 2 contains a subset of three Tivoli product installs (ITCAM, ITSM, and TPM); however, the installations were performed by a different participant who is less experienced in systems management and complexity evaluation.

TABLE II

COMPARING MODELS FROM DIFFERENT TIVOLI INSTALLS.

Group	Total Labor Time	R^2 Calibration	R^2 Extrapolation
1	66.12	0.73	0.32
2	78.54	0.33	0.67

Table II compares the models from the two groups. The difference in the total labor time indicates that the two participants have difference systems management skills. Even if there are only 3 product installations in Group 2, they take longer than the four in Group 1. Comparing the calibration R^2 value also indicates the difference in the complexity evaluation skills —the capabilities to properly define the complexity metrics so as to reflect the complexity in the system in a coherent way.

While Group 1 has a higher R^2 value of 0.73, Group 2 has a lower value of 0.33 due to some inconsistent complexity metric definitions. For example, although the action of launching the ITCAM Install Wizard only takes 3 seconds, its parameter, the path of the Install Wizard, was given a source complexity score of **environmentFixed[5]** instead of **documentationAdapted[3]**. Note that **documentationAdapted[3]** generally indicates less complexity in finding the installation wizard path, based on the instructions in the install manual. In contrast, **environmentFixed[5]** indicates much more complexity involving extensive work by the administrator. Examples include finding the path to a pre-installed executable which could be located at different directories for different systems, or finding the listening port number of a running TCP/IP server, without any hints in the documentation.

It is also interesting to note that although Group 2 has a low R^2 value during model calibration, its R^2 value is much higher during extrapolation when the model is used to predict the labor time in Group 1. This is because even if Group 2's data may not be consistent, the model is still able to find the essential relationship between the complexity metrics and the labor time. Therefore, when the complexity metrics are defined properly (as in Group 1), the model is able to give a good prediction.

F. Are linear Complexity Scales an Issue?

In Section II-B, we had expressed concerns that expressing metric values on a linear scale throughout the model may lead to inaccuracies when building a quantitative model. In order to address this question, we have considered the effect of using a nonlinear scale for Parameter Source Complexity instead of the linear scale of (0, 1, 2, 3, 4, 5, 6) we had defined in [2]. By re-applying our quantitative model to the data from Group 1, a nonlinear scale of (0, 1, 1, 5, 6, 7, 7) can improve the R^2 by only 6%, which seems to indicate that the scale in which qualitative metrics are recorded does not significantly contribute to the accuracy of the model.

V. Related Work

Our model is inspired by the widely successful system performance benchmark suites defined by the Transaction Processing Council (TPC) and the Standard Performance Evaluation Corporation (SPEC). In addition, we have borrowed concepts from Robert C. Camp's pioneering work in the area of business benchmarking [10]. Recent related work in this area is McKinsey & Company's 'Process 360', a benchmark that aims at comparing IT service offerings among different providers [11].

There is increased interest in applying quality management methodologies that have been successfully applied since the 1980s in manufacturing and production to IT services: For example, Six Sigma (in short: 6σ) [12] evolved as a quality initiative to reduce variance in the semiconductor industry, thereby eliminating defects. [13] provides an overview on how to apply 6σ to services.

Related work in the system administration discipline has been carried out with a focus on establishing cost models, which take into account the impact of decisions. The most relevant work is [14], which generalizes an initial model for estimating the cost of downtime [15], based on the previously established System Administration Maturity Model [16].

Other related work can be found in the application of information theory (e.g., Kolmogorov complexities [17]) in order to quantify complexity in workflows [18]. The majority of this work relies on the concept of cyclomatic complexity, which has first been developed in [19].

VI. Conclusions and Future Work

We have described an approach to relate the metrics of a previously developed IT management complexity framework to key business-level performance metrics. We have developed a quantitative model based on a qualitative description of IT complexity parameters together with quantitative calibration data of the IT process and validated its applicability through a small-scale user study. Despite the fact that our user study was done on a small scale, we were able to derive a first set of results that indicate the general feasibility of our approach:

First, by taking only collected complexity measures into account, our model was able to explain 61% of the variability in the time data, which suggests its applicability to forecast trends.

Second, our results have shown that out of a total of ten complexity metrics, two complexity metrics are sufficient to build a quantitative model without jeopardizing its quality. This suggests that there is some potential for simplifying our IT management complexity model, which would increase its consumability.

Third, the dimensions of the scales in which an administrator rates the difficulty he experiences in setting configuration parameters carry relatively small relevance for the accuracy of the model, as they account for only 6%.

Finally, we have found that the skill set of complexity evaluation has a limited impact on the accuracy of the model: While intuition suggests that an administrator less skilled in complexity evaluation would result in a poor data fitting (as evident by that the R^2 error was cut by more than a half), we were surprised to find that the predictive capabilities of our model was not impaired much and it was still able to obtain satisfactory results for other data sets.

These results are just a starting point, and much further work is needed to improve the accuracy of our model. First and foremost, we plan to increase the size of our user study to gain a larger set of data points. In addition, more work is needed to assess whether our IT management complexity model captures the 'right' parameters. So far, it reflects our own experiences with setting up and configuring complex systems. However, more interviews with experts are needed to ensure the model is sufficiently complete. Finally, we envision a future application of our model by including it directly into an IT process modeling tool, so that a process designer can simulate the execution of an IT management process already during the design phase in order to determine whether further process transformations are needed before the process can be deployed in practice..

References

[1] "IT Infrastructure Library. ITIL Service Support, version 2.3," Office of Government Commerce, June 2000.

[2] A. Brown, A. Keller, and J. Hellerstein, "A Model of Configuration Complexity and its Application to a Change Management System," in *Proc. of the 9th IFIP/IEEE International Symposium on Integrated Management (IM 2005)*, A. Clemm, O. Festor, and A. Pras, Eds. Nice, France: IEEE, May 2005, pp. 631–644.

[3] Y. Diao and A. Keller, "Quantifying the Complexity of IT Service Management Processes," in *accepted for publication in: Proceedings of 17th IFIP/IEEE International Workshop on Distributed Systems: Operations and Management (DSOM'06)*, Dublin, Ireland, Oct. 2006.

[4] A. Keller, Y. Diao, F. Eskesen, S. Froehlich, J. Hellerstein, L. Spainhower, and M. Surendra, "Generic On-Line Discovery of Quantitative Models," *IEEE electronic Transactions on Network and Service Management (eTNSM)*, vol. 1, no. 1, Apr. 2004.

[5] L. Ljung, *System Identification: Theory for the User*, 2nd ed. Upper Saddle River, NJ: Prentice Hall, 1999.

[6] J. Zhang and J. Morris, "Process modeling and fault diagnosis using fuzzy neural networks," *Fuzzy sets and Systems*, vol. 79, no. 1, pp. 127–140, 1996.

[7] Y. Diao and K. M. Passino, "Immunity-based hybrid learning methods for approximator structure and parameter adjustment," *Engineering Applications of Artificial Intelligence*, vol. 15, no. 6, pp. 587–600, 2002.

[8] F. E. Harrell, *Regression Modeling Strategies: With Applications to Linear Models, Logistic Regression, and Survival Analysis (Springer Series in Statistics)*. Springer Verlag, 2001.

[9] S. Wold, "Cross-validatory estimation of the number of components in factor and principal components model," *Technometrics*, vol. 20, no. 4, pp. 397–405, 1978.

[10] R. Camp, *Benchmarking - The Search for Industry Best Practices that Lead to Superior Performance*. ASQC Quality Press, 1989.

[11] N. Kaka, S. Kekre, and S. Sarangan, "Benchmarking India's Business Process Outsourcers," *The McKinsey Quarterly*, July 2006.

[12] G. Tennant, *Six Sigma: SPC and TQM in Manufacturing and Services*. Gower Publishing, Ltd., 2001.

[13] B. El-Haik and D. Roy, *Service Design for Six Sigma - A Roadmap for Excellence*. Wiley Interscience, 2005.

[14] A. Couch, N. Wu, and H. Susanto, "Toward a Cost Model for System Administration," in *Proc. 19th Large Installation System Administration Conference (LISA '05)*, D. Blank-Edelman, Ed. San Diego, CA, USA: USENIX, Dec. 2005, pp. 125–141.

[15] D. Patterson, "A Simple Way to Estimate the Cost of Downtime," in *Proc. 16th Large Installation System Administration Conference (LISA '05)*, A. Couch, Ed. Philadelphia, PA, USA: USENIX, Nov. 2002, pp. 185–188.

[16] C. Kubicki, "The System Administration Maturity Model – SAMM," in *Proc. 7th Large Installation System Administration Conference (LISA '93)*. Monterey, CA, USA: USENIX, Nov. 1993, pp. 213–225.

[17] M. Li, *An Introduction to Kolmogorov Complexity and Its Applications*. Springer, 1997.

[18] J. Cardoso, "Approaches to Compute Workflow Complexity," in *Dagstuhl Seminar - The Role of Business Processes in Service Oriented Architectures*, Dagstuhl, Germany, July 2006.

[19] T. McCabe and C. Butler, "Design complexity measurement and testing," *Communications of the ACM*, vol. 32, no. 12, pp. 1415–1425, Dec. 1999.