Deriving IT Configurations from Business Processes: From the Business Process to the Grounded Model

Sven Graupner, Jerry Rolia, Nigel Edwards
Enterprise Systems and Software Laboratory
HP Laboratories Palo Alto
HPL-2007-142
August 22, 2007*

IT Management, business-driven management, SAP, ITSM, ITIL, SOA

Identifying proper business processes, formalizing and documenting them has always been a challenge. Maintaining them in synchronization with changing business needs makes the problem harder. It becomes even harder when business processes are fully or partially implemented in IT systems where changes to the business cause changes in IT configurations.

This paper presents an approach how IT configurations can be derived from business process definitions. Additional information needs to be supplemented to enable the desired function. This information is made available in form of models about application components and their performance characteristics. A series of transformations has been defined between models that constitute the *Model Information Flow*.

The paper presents the overall approach and demonstrates a use case in the second part. The work is part of ongoing joint research between HP Labs and SAP Research.

* Internal Accession Date Only

Approved for External Publication

Deriving IT Configurations from Business Processes

From the Business Process to the Grounded Model

Sven Graupner, Jerry Rolia, Nigel Edwards
Hewlett-Packard Laboratories
1501 Page Mill Rd
Palo Alto, CA 94304, USA
{sven.graupner, jerry.rolia, nigel.edwards}@hp.com

***Abstract*—Identifying proper business processes, formalizing and documenting them has always been a challenge. Maintaining them in synchronization with changing business needs makes the problem harder. It becomes even harder when business processes are fully or partially implemented in IT systems where changes to the business cause changes in IT configurations.**

This paper presents an approach how IT configurations can be derived from business process definitions. Additional information needs to be supplemented to enable the desired function. This information is made available in form of models about application components and their performance characteristics. A series of transformations has been defined between models that constitute the *Model Information Flow*.

The paper presents the overall approach and demonstrates a use case in the second part. The work is part of ongoing joint research between HP Labs and SAP Research.

I. INTRODUCTION

We identify three roles that are primarily involved in creating business applications for enterprises. A business consultant's role identifies and describes participants, processes, and non-functional requirements for qualities of service from the business' point of view. An application platform consultant's role is to customize application artifacts to realize the desired business processes. A solution architect's role then specifies and builds IT systems that support the processes with the desired qualities of service. This may include building new systems, implementing processes in existing systems or integrating processes across systems.

A major challenge in the development of enterprise IT systems is that information captured within the context of one role is often captured informally and communicated informally to those with other roles. As a result building solid and reliable enterprise applications remains to a large extent an art that relies on the knowledge and expertise of the consultants and architects. Published guidelines and established change management practices help to mitigate risks.

On the IT side, the solution architect has to bridge three main areas of concern:

- the business process definitions provided by the business consultant;
- the enterprise applications supporting the desired business processes; and
- the IT infrastructure needed to support the enterprise applications with desired qualities of service.

However, bridging such concerns is a challenge.

In the past, enterprise IT systems were often built as "silos." Multiple IT systems were created and managed in isolation. Humans sometimes integrate such systems by copying, i.e., cut and paste, information from one application to another. Isolation exists at many layers of abstraction within such systems. There are typically dedicated resources for each enterprise application in the IT infrastructure with separate management teams for networks, servers, and storage, isolated applications and change management processes with separate management teams for each application platform, and separate teams considering business process changes.

Recent trends are challenging the traditional way of building enterprise IT systems:

- Service-orientation and service-oriented architectures have become the established principle for modern enterprises [1].
- Faster integration of processes among business partners crossing organizational boundaries has become a competitive factor in business [2].
- The move to consolidated and shared resource environments or utilities is facilitated by next-generation data center technologies [3].

Service-orientation aims to break up silos by turning proprietary applications into services using open standards that support the integration and interoperability between functions and applications in enterprises as services.

The ability to quickly integrate and change processes across organizational boundaries is an increasingly important requirement for competitiveness in modern enterprises. Distributed, collaborative business processes are an approach to support this challenge [4]. IT systems should enable, rather than constrain, the ability to change business processes faster while reducing the costs of change.

And thirdly, it has been well-established that silo'ed IT environments leads to poor utilization or computer resources [5]. More importantly, it is also wasteful of the human resources needed to manage and maintain enterprise applications in data centers, which directly relates to the cost IT imposes on the business. Modern or next-generation datacenters are thus designed around the concept of pools from which computational and human resources are dynamically supplied to enterprise applications based on business needs.

The above trends will significantly impact how business processes and subsequently enterprise applications will be designed, implemented, operated, managed and maintained in future. We anticipate the need for a much more integrated and automated approach that supports the design, configuration, and on-going management for such systems. The approach we propose in this paper aims to increase business flexibility while reducing the costs of enterprise IT systems.

II. PROBLEM STATEMENT AND APPROACH

We aim to provide a more integrated approach to the design, configuration, deployment and management of enterprise IT systems. The approach enables automation while maintaining a separation of concerns among the business, platform, and IT system roles.

There is good reason to keep business processes and logic apart from the details of software platforms and IT infrastructure. Business consultants focus on the value that business processes bring to an enterprise. Application platform consultants focus on how to realize that value using a particular vendor's enterprise application services. IT architects consider how to realize the resulting IT systems from the many alternative implementation methods available to them. We aim to increase integration across roles by relying on more formal specifications of business processes, application platforms and IT infrastructures and enabling the communication of the specifications across roles.

Models are used to capture the specification information about processes, application platforms and IT infrastructures. Model transformations are used to automate many of the mundane steps currently performed by consultants. By automating the steps we aim to reduce the time, costs, and risks associated with change. We do note that not all aspects of design and management can be automated, e.g., writing customized application software code.

The approach taken in this paper assumes several stages of descriptions (models) and transformations between them. Models capture the various aspects of designing IT systems based on functional (business-logic) and non-functional (such as sizing or security or availability) requirements. A chain of models is defined ranging from the business process definition through various stages of refinement to a deployable and finally managed solution. Using a number of separate models reflects the reality that model information comes from different sources and that there are desired boundaries for roles. Connecting models through transformations establishes the linkages to derive the information that allows the automated deployment and management of hardware and software infrastructure.

The overall approach is called the *Model Information Flow*.

III. THE MODEL INFORMATION FLOW

The Model Information Flow [6] represents a number of models that have been identified for capturing the information needed at the different stages for deriving IT configurations from business process definitions. The Model Information Flow represents a flow of information from "the left to the right". This flow of information exists today when business applications are being planned, designed, implemented and managed in enterprises. The difference from today's practice is that the Model Information Flow, formalizes information flows in the form of models from the left to the right as opposed to informal information in the form of informal documents or other means of communication among people. Similarly, model information may flow from right to left to support the management of the enterprise IT system, its application platforms, and business processes.

Figure 1 gives an overview of the models and transformation for the Model Information Flow.

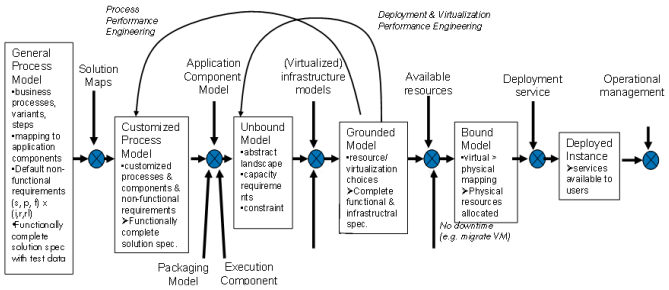

Figure 1: Model Information Flow with transformations.

The information in the Model Information Flow includes the following models.

- The General Process Model is a collection of process descriptions for *best practice* business processes that can be built using available application platforms. A business process consultant selects a subset of these that are appropriate for a customer.
- The business process consultant may customize the processes to better meet the needs of the customer. This may include changing the number or sequence of business process steps or selecting process step variants. Furthermore, the consultant augments the specifications with non-functional requirements, e.g., for performance, security. This is the Customized Process Model.
- Application platform packaging and performance information from enterprise application platform vendors augment the model to indicate which software products and execution platform technologies, i.e, application servers, database servers, are needed to support the required components. This aggregate of information becomes the Unbound Model. It is a set of requirements for the system.
- Infrastructure must then be chosen and configured such that it can support the application design. While in the past this step was largely a matter of choosing and configuring resources (networks, machines, storage), this recently has been changing becoming a matter of exploring configuration choices and creating the virtual resources needed in shared IT environments. This means, resources need to be allocated from pools and virtual resources (networks, machines, storage) may need to be created in them. Exploring design choices based on application requirements and reflecting them in infrastructure designs is becoming more and more relevant. An infrastructure design is the result that is tailored for the Unbound Model in the targeted data center environment. This design is called the Grounded Model.
- The relationship between the enterprise IT system and its allocated resources is described using the Bound Model. As resources are allocated to an enterprise IT system the operating systems, software, and other services are deployed and configured.
- Fully configured resources are associated with the Deployed Model for the enterprise IT system. The

deployed model describes enterprise IT resources that are part of an operational system. These resources are managed with respect to relevant non-functional requirements.

By formalizing model information, some of the transformations can be automated and hence accelerated such that they can be performed and re-performed faster. More specifically, the following models have been identified as constituent models for the various stages of the Model Information Flow. Models are presented in detail in [6]:

- The General Process Model.
- The Customized Process Model.
- The Application Packaging Model.
- The Constraints Model.
- The Application Performance Model.
- The Component Performance Model.
- The Unbound Model.
- The Infrastructure Capability Model (Templates).
- The Grounded Model Design.
- The Grounded Model.
- The Bound Model.
- The Deployed Model.

The Unbound Model acts as the requirements for a system. The Grounded model is a design. The Deployed model describes an operational system. Information from the left hand side, including non-functional requirements, flows through to the right to assist consultants and IT architects and to support the automation of tasks. Information from operational systems can be interpreted with in the context of models from the left hand side. In this way, information also flows from right to left.

Tools such as Aris [7] can be used by business consultants to define and customize processes. Other modeling tools are needed to support other roles. Ultimately, the relationships between models must link business processes and steps to application platform components and IT infrastructure.

IV. Supplementing Business Processes with Non-functional Requirements

A business process specification typically describes the functional requirements for a process. This is not sufficient for deriving an infrastructure configuration. In addition, non-functional requirements must supplement to the process definition to describe the quantity and qualities of the desired system that will execute the process. There are many non-functional requirements for enterprise IT systems including performance, security, and availability.

To narrow the problem space, our current research focuses on performance aspects as part of the non-functional requirements. This section explains how a business process definition from the Customized Process Model is supplemented with performance requirements, which then allow to us to select, evaluate and parameterize a valid IT system configuration design, which is called the Grounded Model. Formalization through models enables the exploration of the design space for the Grounded model using techniques such as Layered Queuing Models (LQM) [8], policy-based design [9] and genetic algorithms [10].

Figure 2 shows a simple business process (SD – “Sales and Distribution”) which is used to benchmark traditional SAP systems. The process consists of 16 consecutive steps. Most steps are actions a user performs via the SAP user interface. These actions cause SAP transactions that are executed by dialog worker operating system processes (Dialog_WPs) within SAP Application Servers. The SAP transactions are application platform components that implement the business process.

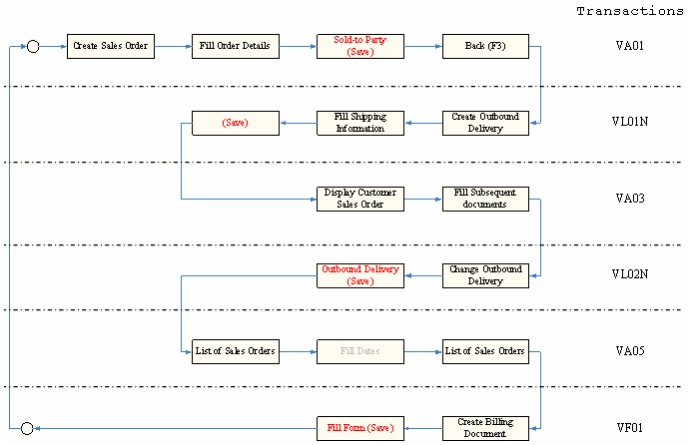

Figure 2: SD Process as example.

Figure 3 shows a schematic view of the SD process as Customized Process Model in the Model Information Flow. The figure shows a UML fragment on the left introducing three classes (AI_Serivce, AI_BusinessProcess, AI_BPStep) and the associations between them. This definition is part of the schema definition of the Customized Process Model (see [6] for the details of this model). The part on the right shows the model of the SD process in a business process editor (the process has been reduced to simplify the presentation).

The editor shows three layers with SDUser being a node in the top layer (“Client”), SDProcess being a node in the second layer (“Process”), and the reduced SD process in the third layer (“Process_steps”).

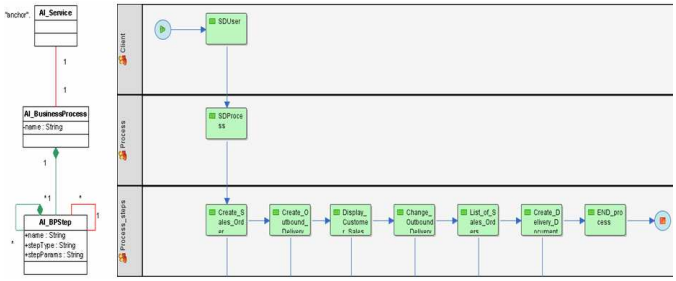

Figure 3: Customized Process Model of SD.

The three layers represent the typical tiers involved in executing the process. A SDUser invokes the SDProcess, which in turn invokes the steps of the process. Steps follow the sequence as shown in the third layer.

Figure 4 shows two kinds of relationships that are occurring in the process definition from Figure 3:

- *invocations* – such as SDUser invokes a process; arrows crossing layers represent invocation relationships, and
- *process logic* – such as CreateSalesOrder is followed by CreateOutboundDelivery; the arrows within a layer represent relationships between process steps.

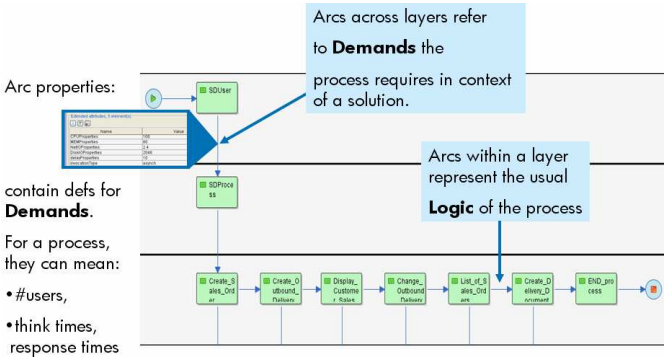

Figure 4: Characterizing and Supplementing Demands.

The business consultant's view only shows the process logic relationships, not the invocation relationships. However, internally, the process definition is expanded to also reflect invocations based on knowledge about the external relationships of a business process. In a simple case, there is at least one entity (the node in the top layer) that will invoke the process, and there is at least one entity (the node in the second layer) that marks the entry point(s) into the process (the nodes in the third layer).

Invocation relationships then provide the placeholders to capture quantitative performance requirements, or demands. Demand requirements can be expressed in various ways. For SAP, the number of (expected) concurrent users plays an important role.

The demand information attached to the SDUser to SDProcess relationship is shown as the large arrow box in Figure 4. It contains few attributes expressing the desired number of users, mean think time, and required mean dialog response time that is expected for the process. Furthermore, the sequencing information for the business process steps is permitted to include loops and branches. For these expected values for loop counts and branching probabilities must also be specified. Together, requirements for users for each role along with the expected values for looping and branching gives the ratio of invocation for the business process steps. These values are entered into an editor by the business consultant as part of the overall requirement gathering process for the Customized Process Model.

It is important to note that non-process related information is not directly visible in the business consultant's view. Information that includes the relationship between business process steps and application platform components is added to the Unbound Model and attached internally to the customized process definition. This information comes from application platform vendors and may be manipulated by application platform consultants.

V. Expanding the Business Process into a Deep Invocation Graph

The expansion of a business process can be continued beyond the layer of business process steps, which is shown in Figure 5. The deep capture expands into the layers of:

- Transactions (initiated by process steps),
- Dialog work processes (Dialog_WPs) which queue and process transactions,
- Databases; and
- Storage layer containing disks.

The static structure of deep invocations is derived from knowledge provided by the application vendors and measurement (SAP or SAP application domain experts in this case). This information is internal and not exposed to the business consultant. An application platform consultant may choose to alter the relationships between business process steps and application platform components, for example to replace an typically used component with a customized component, but we do not consider this in this paper. The business process modeling tool used in our research, however, allows exposing the supplemented information in deeper layers when desired.

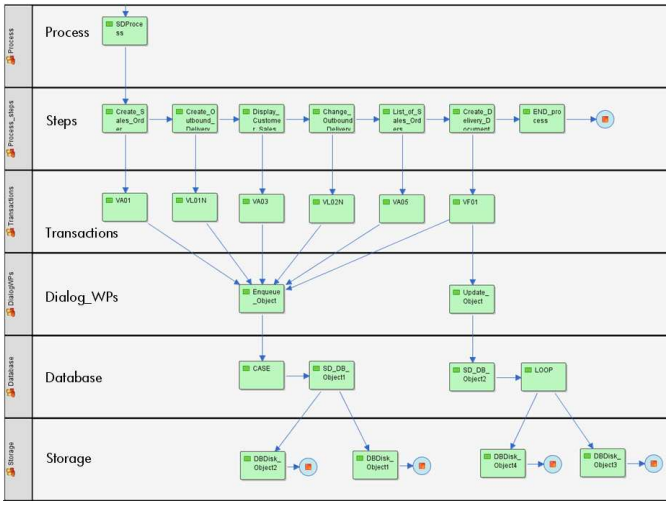

Figure 5: Capturing a deep invocation graph of a process.

The information in the deep invocation graph captures "uses" relationships between application components. At an early stage of configuration design, invocation relationships are merely placeholders to express abstract demand requirements that support the requirements information for the system as expressed in the Unbound Model. At this point, it isn't decided what types of IT system design or resources will be used to support the system so the demands remain as abstract.

VI. Component Performance ModelS: Capturing Components Demands

Component Performance Models are used to characterize detailed invocations among application platform components and demands upon infrastructure. We use measurement based methods to characterize the demands of application platform components on the resource types supported in resource pools. Thus, different choices for resource types yield different demand values. Characterizations of resource usage are stored

in the component performance model repository so that they can be reused. The values are then integrated to create customized process model specific application performance models that support our design exercise.

VII. Design Templates: Describing Infrastructure Capabilities

Design templates are models that describe enterprise IT system variants that can be deployed in an automated manner. As examples, we may have a centralized design that deploys all application platform components and required application and database servers to a single operating system image. A distributed design would have application servers and a database residing in different operating system images. The templates are designed to emphasize support for specific non-functional requirements. For example, a template may include firewalls than another to support a higher level of security.

The templates specify legal ranges for performance related configuration values such as the number of application servers, number of dialog work processes per server, and the concurrency level supported by the database. The choice of values depends on the system's non-functional performance requirements, its use of the application platform components, and the capacity of the resources used to support the system. The specification of these values plus the application platform information from the Unbound Model completes a Grounded Model. This is sufficient information to automatically proceed to Bound and Deployed Models.

VIII. Automated Evaluation of Configuration Designs

In order to perform an evaluation of a configuration design, the three sources of information must be fused together:

- *Customized Process Model* (business process steps and relationships with application components),
- *Component Performance Model* (application component demands on underlying system),
- *Design Templates* (IT system design including specific resource types).

Figure 6 shows the process of evaluating configuration designs based on the models discussed.

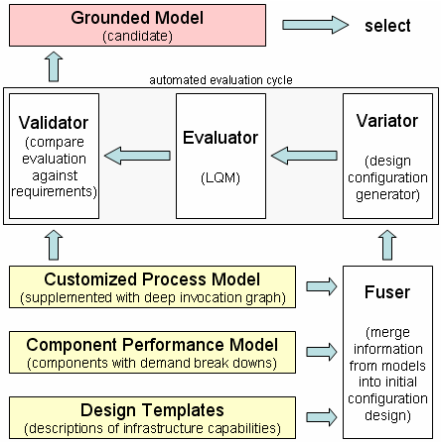

Figure 6: Process of deriving Grounded Models.

The *Fuser* reads the information from the three sources and merges it into an initial configuration design.

The *Variator* then consumes this initial configuration design and triggers the evaluation of the design. The Varitator will also identify possible permutations of configuration choices altering the initial configuration design and feed them into evaluation as well.

The *Evaluator* evaluates a configuration design. A Layered Queuing Model (LQM)-based tool [11] named the Method of Layers (MOL) tool is used for that purpose. The result of the evaluation is list of latencies expected at the different components, e.g. at the database. Those latencies then can be compared against requirements leading to the exclusion of the evaluated design.

The *Validator* then can compare those latencies against requirements from the Customized Process Model.

If all the requirements are met by a design, LQM also provides information about initial numbers of instances of components such as numbers of application server instances needed to support the design. Those numbers then are supplemented into the evaluated design template turning it into a candidate for a Grounded Model which potentially could be deployed when it is selected. Initially, we consider Grounded Models produced by this method as proposals to solution architects enabling them to make better founded decisions about infrastructure configuration choices. However, in future these Grounded Models may flow directly into deployment systems where their actuation can be scheduled.

Since the Model Information Flow is build on formalized models, interactions between its components can be automated. The automated evaluation of configuration designs is an example of that. The design space of our current research prototype is still limited (see the example in section IX). However, we expect in future that potentially large design spaces can be explored and automatically evaluated by this method.

IX. Layer Queuing Model-based Evaluation

LQMs were developed [12], [13] to study distributed software systems that share hardware resources such as processors, disks and network elements. They are extended Queuing Network Models (QNMs) [14] that take into account both demands at hardware resources and visits and queuing between software components. Like QNMs, they operate at a high level of abstraction and are suitable for comparing system designs and for capacity planning. The models are best suited to systems with many clients, for example, enterprise application systems.

Figure 7 shows information from the Unbound Model that is captured in a LQM. The figure shows a user's interaction with the system and its use of an application platform component, i.e., a SAP Transaction dialog step, its use of application server and data base server operating system processes and their abstract use of hardware resources. The LQM captures the aggregate of this information over all roles and all business process steps. In addition the LQM includes the number of users for each role, mean response time requirements, expected think times, and expected values for

loop counts and branching probabilities. These are all aspects of non-functional requirements from the Customized Process Model.

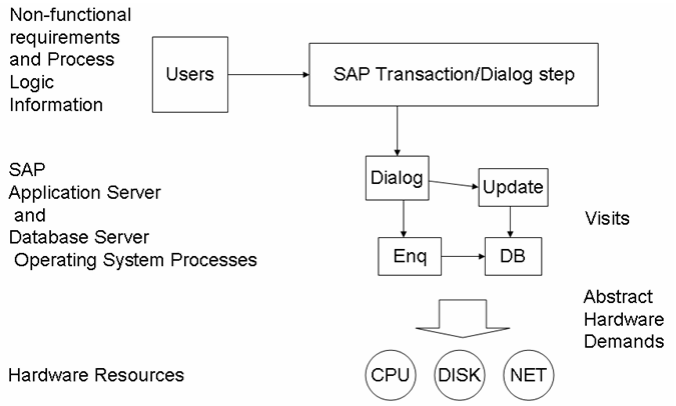

Figure 7: Unbound Model information for the layered queuing model.

Figure 8 shows design information from centralized and distributed enterprise IT design templates that are included in a LQM. The template supports any Unbound Model that makes use of infrastructure described by the template.

The LQM further includes capacity information for the resource types used by the template and estimates for the demands on the resource types from the component and application performance models.

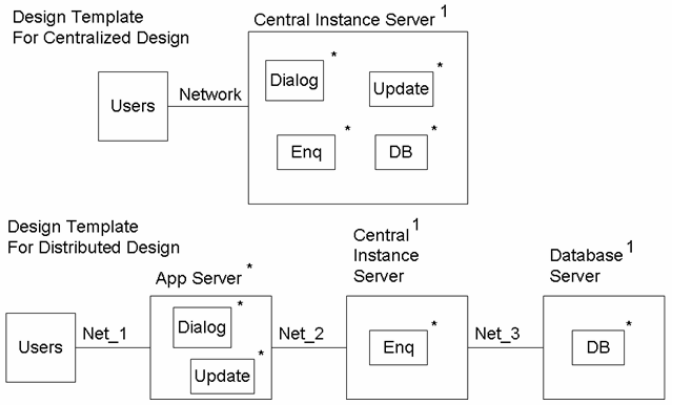

Figure 8: Design Template information for the layered queuing model.

In the LQM, Clients interact with Dialog Work processes, Dialog Work processes interact with Enqueue and Update Work processes, Enqueue and Update Work processes interact with processes in a Database Server. All of these make use of hardware resources such as CPUs, disks, and networks.

The solution technique for LQMs is called the Method of Layers [11]. In general, the layered models are partitioned into a sequence of two level sub-models that are solved iteratively using variants of Mean Value Analysis (MVA) [15]. For example, a first sub-model considers users interacting with Dialog Work Processes. A second sub-model considers Dialog Work Processes interacting with Enqueue and Update Work processes. The performance results of one model affect the input parameters of the others. The iterations converge to a fixed point when the sub-models have consistent performance behavior. MVA residence time expressions have been modified to reflect synchronous, asynchronous, and multi-threading aspects of software interactions.

We construct an LQM for each alternative IT system design template. The template aspect of an LQM model includes information from the networking level up to the operating system process level and is invariant across customer systems. It is reused for each system under study. Business process information from the customized process model tailors the model to reflect a particular customer's needs in terms of business process steps, interactions among application platform components, and demands upon devices. A customer's performance requirement is expressed as a number of users for each user role and a mean response time and think time requirement for user interactions with the system. An optimization subprogram repeatedly tunes and solves the LQM to satisfy the customer requirement. The technique reports an estimate for the number of servers and number of operating system processes for each server for each tier in the system. This information completes the grounded model for the system.

X. Evaluating the Centralized Configuration for the SD Process

As a case study we consider the evaluation of a SAP system for a centralized system design. Table 1 shows several specifications for non-functional performance requirements along with corresponding performance configuration values. In all cases the mean customer think time was 10 seconds per dialog step as is typical for the SD benchmark.

Num User	Mean Dialog Resp. Time Requ. mSec.	Num of Dialog WP	Num of Update WP	Num of Enqueue WP	Num of DB Process
1000	2000	8	2	3	12
1775	2000	83	3	16	102
1775	2500	55	2	4	61

Table 1: Performance Value Results for Centralized Application Design Template.

Table 1 shows that for 1000 SD users and a mean dialog response time requirement of 2000 milliseconds the enterprise IT system under study required 8 Dialog Work processes, 2 Update Work Processes, and 3 Enqueue Work Processes. The database had to be able to support up to 12 concurrent transactions. The resource type we considered was able to support up to 1775 users with a mean dialog response time limit of 2000 milliseconds. For that scenario, we require significantly more Work Processes and concurrency at the Database. This suggests that the resource type must have significantly more memory (e.g. nearly 10x) than for the 1000 user scenario. By reducing the mean dialog response time requirement to 2500 milliseconds the number of Worker Processes and hence memory requirements drops considerably.

For a given set of non-functional requirements the LQMs enable us to report on how much capacity is needed for each design template alternative. The template that best satisfies non-functional requirements with the lowest requirement for capacity is recommended to the IT solution architect.

The IT solution architect interacts with the LQM evaluation tools though an Excel spread sheets. The input into the LQM spreadsheet is automatically filled in using the process shown in Figure 6. Figure 9 shows the spreadsheet that is presented to the IT solution architect.

	A	B	C	D	E	F
1	*To Invoke Optimizer -> Tools/Macro/Macro/Do_optimize*					
2	**Goal**					
3	NumberOfSDUsers	**1000**				
4	SDUserDialogMeanResponseTimeMs	**2000**				
5						
6						
7	**Factor**	**TempSpace**	**LowValue**	**HighValue**	**RecommendedValue**	
8	NumAppServer	1	1	1	1	
9	NumDialogProcessesPerAppServer	83	2	500	83	
10	NumUpdateProcessesPerAppServer	3	1	500	3	
11	NumEnqueueProcessesInCI	16	2	500	16	
12	NumDBThreads	102	2	500	102	

Figure 9: The LQM result sheet.

Interaction with the evaluation results also allows to change parameters such as number of users for which a configuration is sized and re-running the evaluation. Designs can thus be changed and re-evaluated at a much faster pace then implementing them as test systems and exploring configuration choices in a real system.

XI. Related Work

A large body of work exists in the domain of business process design [16], business process automation [17] and business process management [18]. Business process languages have widely been proposed and used, particularly in context of web services [19]. Formal work on business processes has been done in the academic domain [20]. More recently, collaborative business process environments have been proposed [4].

On the other hand, IT automation and automated IT systems have become relevant trends in recent years in the IT industry. Initiatives such as Adaptive Infrastructure or Autonomic Computing reflect the situation enterprise customers face in growing complexity in IT systems along with increasing cost. Industrial research in this field reflects the need to support management of IT systems and ideally automate it [21], [22], [23]. Integrated Service Management (ITSM) aims to streamline and standardize processes in overall IT management [24] addressing not only the technical , but also organizational aspects in order to make IT management more efficient in enterprises.

However, although both worlds of business processes and IT systems are closely linked today in enterprises, they still remain widely disconnected conceptually organizationally and technically. Different groups of people with different background and skills work in the different domains.

The resulting fracture between the business layer and the IT layer already causes substantial pain today, and it is projected to become even more painful in future as the interactions between businesses continue to evolve. Only few intersection points have emerged between the domain of business processes and enterprise applications and IT systems such as business process monitoring or business-driven management [25].

Today, the gap between the two layers must be bridged by human expertise that is involved in planning, designing, implementing and managing the IT systems to support all the processes enterprises depend upon today. Substantial cost is involved. But more importantly, IT systems and their limited ability (or better disability) to change have become a restrictive factor for business innovation in enterprises. Changing a process can takes months in preparation and planning alone; followed by again months for designing, implementing and testing systems on the IT side before the desired change can become effective in the production system.

Easing and supporting the complex and long-term processes that are involved in accommodating change in business in enterprises has thus become a major differentiator vendors of enterprise software and systems aim to provide to their customers. Establishing a capability to derive IT configurations from business processes is part of those efforts. Once this capability is established, changes to IT system configurations can be automatically produced as changes in business processes occur. These changes can be evaluated and tested as at the stage of design candidates much faster and efficiently than implementing them in form of test systems. Introducing this methodology into the way how IT systems are designed, validated and implemented, is expected similar synergies and advantages as other industries have achieved with migrating to computer aided design, test and implementation technologies.

XII. Summary

The work presented in this paper aims at an intersection point between the business process domain and the domain of IT configuration design by establishing the linkage between both domains in form of the *Model Information Flow*.

The goal is to derive IT configurations from business configurations. In order to enable this function, additional information must be supplemented in form of the Model Information Flow. This information formalizes knowledge IT solution architects use to properly plan, design and size and implement enterprise applications. Information includes the structure of application stacks and performance characteristics of application and infrastructure components. All this information is then fused together using transformations.

A particular evaluation technique LQM has been presented within the context of the Model Information Flow that allows to compute configuration parameters such as numbers of work processes to meet non-functional performance requirements provided in form of numbers of users and expected mean response time. A bounded space of design choices can automatically be evaluated. The final result is then used to configure the chosen infrastructure design.

Future work will also consider other domains of non-functional requirements such as security and manageability. Both aspects need to be factored into IT configuration designs similarly like the actual functional building blocks with the desired performance properties. Manageability, for instance, requires monitoring. Monitoring requires a monitoring infrastructure to be factored into an IT design for collecting and processing monitored data and interacting with a monitoring or management system. Once monitoring is available, results can be compared against the estimates made in designs and the results approximated by the design evaluation.

Other future work will also focus on finer-grained capture of the internal application structure in models of the Model Information Flow as well as the fine-grained performance

characterization of application and infrastructure components for a wider range of applications than the demonstrated SAP SD process that was used in this paper.

REFERENCES

[1] Krafzig, D. Banke K. Slama D.: *Enterprise SOA: Service Oriented Architecture Best Practices*, Prentice Hall, 2005.

[2] Scheer, A.W., Abolhassan, F., Jost, W., Kirchmer, M. (Eds.): *Business Process Automation*, ISBN 3540207945, Springer Verlag, 2004.

[3] Hewlett-Packard, *Next Generation Data Center*. http://www.hp.com.

[4] Sadiq, W, Sadqi, S., Schulz, K.: *Model Driven Distribution of Collaborative Processes*, IEEE International Conference on Services Computing. SCC 2006, Chicago, USA. Sep 2006.

[5] Rolia, J., Zhu, X., Arlitt, M., *Resource Access Management for a Utility Hosting Enterprise Applications*, 549-562, IM 2005.

[6] Belrose, G., Brand, K., Edwards, N., Graupner, S., Rolia, J., Wilcock, L.: *Business-driven IT for SAP – The Model Information Flow*, Second IEEE/IFIP International Workshop on Business-driven IT Management BDIM 2007, Munich, Germany, May 2007.

[7] IDS Scheer: *Aris*, http://www.ids-scheer.com.

[8] Herzog, U., Rolia, J.: *Performance Validation Tools for Software / Hardware Systems*, Performance Evaluation 904 (2001), 1-22.

[9] Graupner, S., Sahai, A.: *Policy-based Resource Topology Design*, 5th IEEE/ACM International Symposium on Cluster Computing and the Grid (CCGrid 2005), Cardiff, UK, May 9-12, 2005.

[10] Vose, M.D.: *The Simple Genetic Algorithm: Foundations and Theory*, MIT Press, Cambridge, MA, 1999.

[11] Rolia, J., Sevcik, K.C.: *The Method of Layers*, IEEE Trans. on Software Engineering, Vol. 21, No. 8 pp. 689-700, August 1995.

[12] Woodside, C.M., Neilson, J.E., Petriu, D.C., Majumdar, S.: *The Stochastic Rendezvous Network Model for Performance of Synchronous Client-Server-like Distributed Software*, IEEE Transactions on Computing, Vol. 44, 20-34, 2995.

[13] Rolia, J., Sevcik, K.C., *The Method of Layers*, IEEE Transactions of Software Engineering, Vol. 21(8), 689-700, 1995.

[14] Lazowska, E.D., Zahorjan, J., Graham, G.S., Sevcik, K.: *Quantitative System Performance Using Queuing Network Models*, Prentice Hall, Englewood Cliffs, NJ, 1984.

[15] Reiser, M.: *A Queuing Network Analysis of Computer Communication Networks with Window Flow Control*, IEEE Transactions on Communication, 1201-1209, 1979.

[16] Scheer, A.W.: *Business Process Design*, Springer; 3 Edition, September 22, 2006.

[17] Scheer, A.W., Abolhassan, F., Jost, W., Kirchmer, M. (Eds.): *Business Process Automation*, ISBN 3540207945, Springer Verlag, 2004.

[18] OMG: *Business Process Management Initiative*, http://www.bpmi.org/.

[19] Leymann, F.: *Business Process Execution Language for Web Services (BPEL)*. Leymann, F.: *Web Services Flow Language (WSFL)*.

[20] Aalst, Wil M. P.: *Formalization and verification of event-driven process chains*. Information & Software Technology 41(10): 639-650 (1999).

[21] Singhal, S., Graupner, S., Sahai, A., Machiraju, V.: *Quartermaster – A Resource Utility System*, IM 2005, Nice, France, May 15-19, 2005.

[22] Graupner, S., Cook, N., Coleman, D.: *Automation Controller for Operational IT Management, Integrated Management*, IM 2007, Munich, May 2007.

[23] *Smart Framework for Object Groups*, http://www.smartfrog.org.

[24] *IT Service Management, The ITIL and ITSM Directory*, http://www.itil-itsm-world.com.

[25] Claudio Bartolini, Mathias Sallé: *Business Driven Prioritization of Service Incidents*. DSOM 2004: 64-75.