Towards simulation-supported enterprise architecture management

Sabine Buckl[1], Florian Matthes[1], Wolfgang Renz[2],
Christian M. Schweda[1], Jan Sudeikat[2,3]

[1]Lehrstuhl für Informatik 19 (sebis)
Technische Universität München
Boltzmannstr. 3
85748 Garching
{buckls,matthes,schweda}@in.tum.de

[2]Multimedia Systems Laboratory (MMLab)
Fakultät Technik und Informatik
Hamburg University of Applied Sciences
Berliner Tor 7
20099 Hamburg
{wr,sudeikat}@informatik.haw-hamburg.de

Abstract: Enterprise architecture management is based on a holistic view on the enterprise addressing business and IT aspects in an integrated manner. EA management is a process to manage the complexity of the overall architecture and to improve the alignment of business and IT. In order to achieve these goals, it is necessary but not sufficient to manage the static complexity that arises from dependencies between the elements of the EA, like goals, organizational units, business processes, business applications, and IT infrastructure elements. Performance, stability, and scalability can only be analyzed, modeled, and controlled, if static EA models are enriched by appropriate abstractions to capture the dynamic complexity of the EA understood as a socio-technical system of interacting (sub–)systems. This article identifies possible techniques to address dynamic complexity in EA. The potential benefits of system simulations are discussed and the derivation of apropriate simulation models is exemplified. A key observation is the fact that EA management is a iterative evolution process, where each iteration only changes a small fraction of the EA. It is therefore possible to automatically derive model parameters required for the simulation of the future architectures from an analysis of the dynamics of the current architecture.

1 Motivation

In large enterprises, the application landscape, as the entirety of the employed business applications [Wit07], is an important asset, which is a critical support factor for business and

[3]Jan Sudeikat is doctoral candidate at the Distributed Systems and Information Systems (VSIS) group, Department of Informatics, University of Hamburg, Vogt–Kölln–Str. 30, 22527 Hamburg, Germany

a costly investment constantly demanding maintenance operations. Consequently, managing the application landscape has not only recently become a challenge, current enterprises have to address. Especially in the larger context of enterprise architecture (EA) management, the alignment of the enterprise's business and IT, forms the focal point. In the light of this increased interest from practitioners, different approaches to facilitate EA management have been developed in academic research [FAW07, Fra02, Lan05, Wit07], by practitioners [Der06, EHH$^+$08, Kel07, Gro05], and by tool vendors [MBLS08]. Nevertheless, the approaches concentrate on structural aspects of the EA or the landscape respectively – such as interconnections between business applications.

In contrast, the EA and the application landscape in special form highly dynamic systems, with their behavior being far more complex than their topological structure is likely to indicate. Such *dynamic complexity* is a widely known and accepted fact in many management disciplines, of which first references date back to the 60^{ths} of the last century (cf. the *Forrester effect* [For61]). Concerning application landscapes or EAs, such considerations have not yet been undertaken, although some bordering fields from EA management, e. g. business process management, have already established means and methods for addressing and evaluating behavioral complexity, e. g. business process simulation [JVN06, LM04]. Thereby, the simulated concepts are considered on a high level of abstraction, omitting detailed internal descriptions in favor of a more *phenomenological* treatment of the subject. We regard the establishment of complementary high-level simulation methods for EAs a next important step towards a more mature management discipline and present an approach to this area alongside an application example in Section 4 of the article. Preceding Section 2 provides the foundation for the approach. Section 3 supplements some more elaborate considerations on the EA as a dynamic system. In conclusion, Section 5 gives indications on future research topics in this area.

2 Related work

The simulation approach presented in this paper targets two bordering fields – EA management and simulation techniques. An introduction to both areas is subsequently given, and their impact on the presented approach is outlined.

2.1 EA management

The topic EA management is an important issue in academia as well as in practice for several years now. One of the first papers targeting this field dates back to 1992, when Sowa and Zachman developed a framework for information systems architecture [SZ92]. Since that time, the number of publications concerning EA management has grown constantly [LW04], although no commonly accepted definition of the terms EA or EA management exists. Nevertheless, the various definitions available center around a common concept – a holistic view on the enterprise. Thereby, the areas of considerations may be

different, but mostly range from infrastructure to business or strategic aspects. Figure 1 shows the different layers and crossfunctions of an holistic EA management approach according to [MBLS08].

The color coding of Figure 1 indicates the variability concerning the involvement of information from specific layers or crossfunctions according to the different definitions of EA. While it is common, that information from all layers and cross functions is of importance in the context of EA management, different approaches vary widely in their definition to which level of abstraction this information should be detailed (see e. g. the approaches of [Fra02] and [BELM08]).

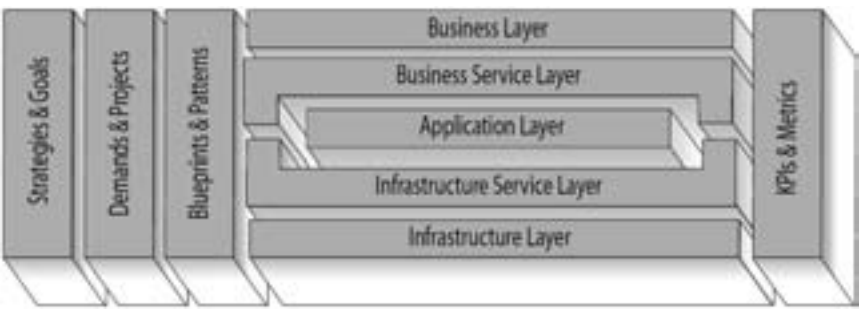

Figure 1: Layers and cross functions of an holistic EA management approach [MBLS08]

The elements of an EA as alluded to above are subject of the process of EA management. According to [MBLS08] EA management can be defined as *a continuous and iterative process controlling and improving the existing and planned IT support for an organization. The process not only considers the information technology (IT) of the enterprise, also business processes, business goals, strategies etc. are considered in order to build a holistic and integrated view on the enterprise. The goal is a common vision regarding the status quo of business and IT as well as of opportunities and problems arising from theses fields, used as a basis for a continually aligned steering of IT and business.* One central task of EA management is the management of the application landscape [Lan05, Wit07]. As managing a complex asset as the application landscape is likely to involve a large number of people with different backgrounds (e. g. business process managers, project managers, or business application owners) communication is a major management issue. Therefore, graphical models called *software maps* [LMW05, BELM08] (as the one shown in Figure 2), have proven to be a valuable means of communication in this context. A real-world exemplary visualization[1] focusing on business application interdependencies and their distribution is shown in figure 2.

In addition to visualizations, metrics can be used to bridge the communication gap between the stakeholders from business and IT [AS08, LS07, LS08, Lan08]. Thereby, metrics build a more objective basis for decision making, helping the manager to not simply rely on his *gut feel*. [LS08] and [Lan08] demonstrate the applicability of metrics in the context of application landscape planning through comparing planned landscapes in respect to failure

[1] Due to reasons of confidentiality the figure, which is taken from a EA practitioner, is made unreadable.

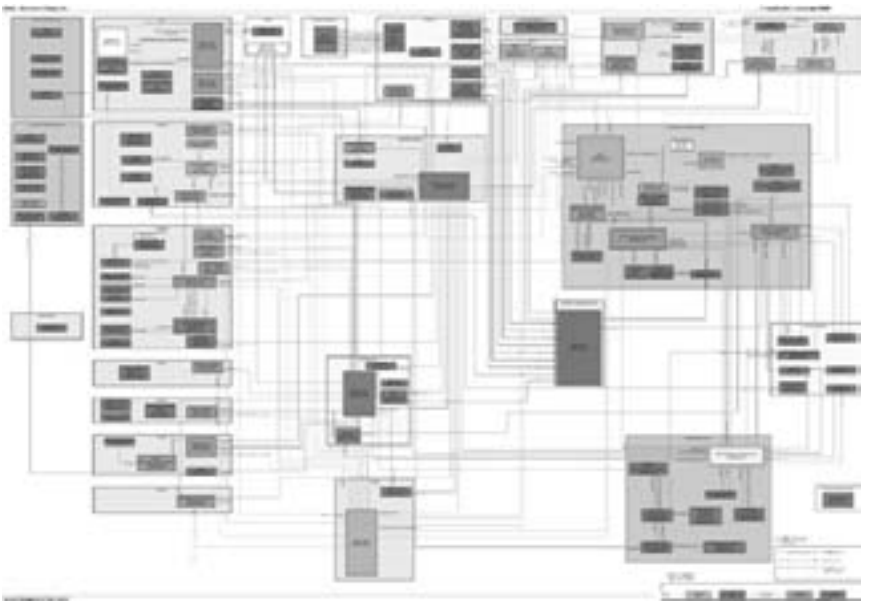

Figure 2: Software map focussing on business application interdependencies

propagation. Notwithstanding the importance of metrics concerning structural dependencies in the application landscape, also measurable properties exist, which arise from the interaction in these highly dynamic systems.

2.2 Simulation techniques

The only means to assess this dynamic complexity are system simulations (cf. [BF04] for an overview of simulation approaches), which therefore is a necessary prerequisite for measuring dynamic system properties. The observation of the timely development of system properties is supported by numerous simulation techniques ranging from purely mathematical approaches, e.g. *equation based modeling* (EBM), to formal modeling approaches, e.g. petri-nets [vdA03] and process algebra [Fok00] as well as object- and agent-oriented simulation methodologies [PSR98]. Mathematical and formal models are simulated by iterative evaluation, while the computational simulation models rely on the emulation of constituent system elements [PSR98]. In the following, we briefly outline the applicability of some of these models for the simulation of EA.

A prominent purely mathematical simulation technique is the *queueing theory* (see e. g. [Tij03]), a widely accepted and used means for deriving performance measures, e. g. processing time, in systems with a continuous random stream of incoming requests [vdA03, BLL+94]. Such a situation can be found at multiple occasions in the context of EAs, e. g. concerning business process execution in the back-office. The steps forming these pro-

cesses are thereby executed by business applications – their throughput performance can be computed using queue based simulation models. Such models have been successfully applied on business level (see e. g. [KB03]) or on infrastructure level concepts of the EA as well as on intra-application aspects, like load balancing mechanisms. Contrastingly, up to now, no attempt to use these queueing based simulation techniques on a more holistic model of the EA has been undertaken. This might be due to the fact, that on the level of the whole EA load information is often only known on a rather qualitative level, lacking precise quantitative information, which would be needed for queueing models. Additionally, an embracing, enterprise-wide queueing model is likely to fail for a complexity problem.

Formal simulation techniques, e.g. petri-nets and process algebra, are particularly attractive to model concurrent activities and the resulting macroscopic, dynamic system behavior. Petri-nets [vdA03] describe systems as graph structures, where system states are represented as markings of tokens spread over nodes. This modeling notion complicates the description of component data (*business objects*) and its change during state transitions, quickly calling for more sophisticated methods like colored petri-nets [Jen96]. Process algebra models [Fok00] extend the former approach by supporting compositionality, but as well adopt a purely behavioral stance toward component behaviors. Therefore, both modeling approaches allow to express the dynamic system behaviors, but their abstraction level demands additional effort to interpret the obtained results and relate them to the represented EA concepts.

Object- as well as agent-oriented simulation models enable fine-grained views on system components to resemble the resulting macroscopic system dynamics. In these approaches the simulation model consists of collections of passive or pro-active elements, respectively. Agent-based models are attractive to represent EA aspects as they support modeling systems as organizational structures of autonomous components [MY04] that decide locally on their interactive behavior. This abstraction level supports the representation of intelligent architectural elements that decide themselves to engage in computations, e.g. trigger business processes executions. Therefore, complex local rules, which control interactive computations, i.e. the realization of business processes, can be directly encapsulated to observe their effect on the resulting system behavior. In [PSR98], agent-based simulation models are distinguished from mathematical approaches. It has been argued that the ability to directly represent the element's internal computations in agent models, facilitates the transfer from simulation models to the represented systems. Particularly interesting in the context of EA simulations is the ability to enable experimentation by *what-if* games, e. g. via application landscape scenarios. The ability to directly model system components and their local computations can enable enterprise architects to alter local strategies and observe their impact on the resulting global system behavior. Moreover, the direct mapping between simulation models and the simulated subjects facilitates the implementation of the gained insights, i.e. modifications of the represented EA element. Due to these benefits, agent–based simulations are preferable to rigorous mathematical / formal simulation approaches. We expect that the local decisions of EA components can be mapped to reactive, rule–based reasoning mechanisms, which facilitates the utilization of sophisticated simulation environments [IGN05] that ensure the scalability of the simulation approach for large scale EA.

3 Dynamic system aspects of EAs

Application landscapes, as part of EAs, exhibit a multiplicity of relations within the set of interconnected business applications and components. These relations give not only rise to *structural* complexity but, according to the dynamic properties of the underlying dependencies, cause *dynamic* complexity. In system dynamics, analytic and constructive issues are considered, which are explained here because they will influence the interpretation and validity of statements about dynamic properties of a system.

System dynamics research coined the term *dynamic complexity* to describe the intricate difficulty to assess the long-term behavior of dynamic systems. This notion of complexity is distinguished from the complexity that rises from the combinational number of elements and their relationships. It describes the rise of complex, typically dysfunctional behavior that evolves from subtle cause and effect structures. When the long-term effects of short-time interventions are not obvious, this complexity can be observed in comparatively simple systems with low structural complexity.

In dynamic system analysis it is well known that the behavior of system components can depend on the very details of the imposed conditions. To be specific, constant work load or repeated transient peak load conditions with identical average work load can lead to dramatically different response behavior and throughput. Such effects often are not recognized, if the simulation model does not cover every detail, which at modeling time cannot be identified as crucial. System architects are trained to identify components of concern and the possible appearance of conditions causing unwanted behavior and provide means to stabilize the system properties. Nevertheless, in many cases such unwanted behavior is not discovered at system design time and is often difficult to detect in the running system because of its *dynamic complexity* induced by its complex internal dynamics as a response to simple use-cases.

Since the subtleties of causal relations among system components can lead to dysfunctional system behaviors (e. g. discussed in [Mog05]), the architects of EA face the challenge to infer the dynamic system properties from the structural properties of the conceived EA. We propose the use of system simulations to facilitate the examination of enterprise-wide dynamics and the anticipation of effects of landscape modifications. Architects will use these simulations to examine both the *operational* properties of EAs, e. g. performance measurements, and strategic, non-functional properties, as scalability or robustness, which are influenced by global architectural decisions.

4 Exemplifying our simulation approach

Simulation techniques are commonly accepted means for getting insights into the dynamics of a system in many engineering disciplines. There actually exists a multitude of reasons, why simulation is employed, of which an overview is given e. g. in [Wik08]. Especially the following ones strongly apply in the context of simulating EAs or application landscapes respectively, namely:

- Analyzing the real application landscape's behavior in detail would be very costly and resource-consuming.
- Analyses of the behavior of future application landscape's cannot be performed, as setting up them in an experimental environment is not possible due to the expenses connected.
- Changing execution parameters in the real application landscape for reasons of analyses is not appropriate, as the effects might cause severe business incidents.

Creating a model suitable for simulation of the EA or the application landscape can therefore be considered a sensible way to approach the aspect of dynamic complexity. Such an approach is necessarily complemented with measures to assess certain aspects of the landscape and present them in a condensed form, e. g. by using application landscape metrics [Lan08, LS08].

Subsequently, we present our approach to EA simulation. Thereby, a real world problem of a large bank is used to briefly exemplify the steps to be taken. The large bank operates an application landscape containing a backend, which is purely host-based, i. e. it relies on mainframe systems, on which the core business functions are executed, while a distributed presentation logic exists. With the ever changing business environment on the one hand and the decreasing number of available software engineers skilled in host programming languages on the other hand, a major change in banking application landscapes is at sight. A transition is likely to lead from centralized host architectures to distributed (potentially service oriented) architectures. Nevertheless, this change of paradigm also brings along important uncertainties, especially concerning the latency and failure propagation of the revised application landscape. To illustrate the distribution issue, Figure 3 shows a simplified example of a host-based realization of a business process execution.

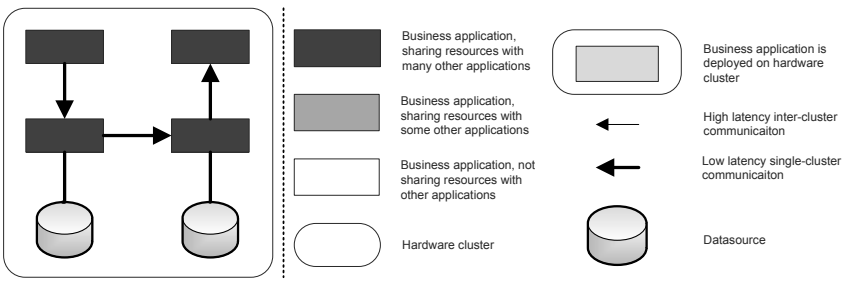

Figure 3: Cutout of the host-based landscape involved in the execution of a business process

Latency and failure propagation on the level of business process execution are clearly dependent on the distribution topology of the new landscape. This topology lays the ground for the complex dynamics, which can lead to reduced throughput and increased latency. Consequently, a company would look to achieve optimal performance properties for the landscape, by choosing the *right* topology. Nevertheless, finding out, what is *right*, must

be considered a complex task, for which the enterprise architects have not had the opportunity to develop a *gut* feel. This lack of experience is addressed by us via metrics indicating e. g. average latency based on simulated dynamics of the application landscape. Thereby, the metrics could be applied to different scenarios for future application landscapes, either created by an enterprise architect or machine-created ones randomly distributing the business applications on systems. After simulating the scenarios' dynamics and evaluating their performance indicators, the architects could choose the future landscape optimally fullfilling their performance requirements.

Alongside this example, we introduce the steps of an continuous and iterative process, supporting the simulation of various aspects of an EA. Thereby, the system of which different elements are simulated, is the EA or respective parts thereof, e.g. the application landscape. Due to reasons of brevity the example concentrates on the issue of latency as introduced above. The steps to be taken are:

1) Gather *concerns and objectives:* At first the pain points, which should be addressed by the simulation, have to be identified, e. g. by interviewing the respective stakeholders and gathering their concerns[2] regarding the system. These concerns have to be operationalized to objectives or key performance indicators (KPIs) in order to support the measurement of the target achievement.

> Currently, the bank operates a host-based backend system for supporting its backoffice banking services. Driven by the lack of experienced host-programmers, the bank has chosen to migrate its backend to a distributed Java-environment. Thereby, questions like the following have to be answered prior the migration:
>
> - How does the distribution affect the overall system performance, especially concerning the execution time of LUWs (latency)?
> - What complex behavior might arise, e. g. regarding failure propagation in partially asynchronous environments? Consider a situation, in which a *fast* system having an incoming queue fails and is brought back to operation after a certain time. May it cause subsequent systems to break down due to too much load in little time?

2) Define *objects of observation:* In order to determine the parts of the system, which are relevant for the simulation, the objects of observation and their dependencies have to be identified. In addition, KPIs of the system have to be related to the objects of observation.

> Business processes are supported by business services, which are realized by business applications utilizing infrastructure services. Currently, the applications supportive for a process are executed on a single host, thus neither distribution of data nor network communication is employed.
>
> The performance analysis in the scenarios mainly targets the processing times for the business processes to be executed. Therefore, the *average processing*

[2]The terms *concern* and *stakeholder* are used here in accordance to the definitions provided in the IEEE Std 1471-2000 [IEE00].

time per business process request and the *standard deviation* of processing time per business process can be considered to be suitable KPIs.

3) Identify classes and develop respective *information model:* The objects identified in the preceding step have to be classified and their relationships to each other have to be conceptually modeled. This leads to an information model, which represents the influential constituents of the landscape and their relationships to each other according to the simulation scenario addressed.

Figure 4: Simplified information model for the banking example

An information model for the banking example has to consider the business processes, the supportive service-providing business applications, and the infrastructure services, which are utilized by the applications. Additionally, the ordering of business service calls in the execution of a business process has to be taken into consideration as well as the information flows between the business applications. Figure 4 provides a simplified information model for the banking example, in order to indicate the level of granularity on which information is needed for simulation. The concepts introduced therein should be self-explanatory and are hence not detailled here.

For supporting simulation, this static aspects have to be complemented with further dynamic information, e. g. concerning the distribution of business process requests over time. Additionally, means to model the resource usage of an application in executing a service request have to be incorporated in the information model, e. g. via in-application cause-effect modeling facilities. Further, the distributed environment calls for ways to model different types of inter-application dependencies:

Communication dependencies two or more business applications communicate in the execution of one business process either synchronously

(mostly) or asynchronously (seldom).

Resource dependencies two or more business applications from different business processes compete for the same resource, e.g. CPU.

Data dependencies one or more business applications are triggered by events raised by data changes via another business application.

4) Choose appropriate *simulation technique:* Before starting the simulation process, an appropriate simulation technique has to be chosen. As introduced in Section 2, a multitude of different simulation methodologies exists, of which the one best suitable should be taken according to the characteristics of the simulation and the system under consideration.

Purely equation based models could be used to simulate the dynamics of the described system in detail. Nevertheless, analytical solutions may not be easy to find due to the generality of the assumptions and the high level of abstraction. Further, especially resource dependencies cannot be modeled easily in those mathematical models.

Formal simulation tools, as petri-nets or process algebras, may seem appealing in this context, although their main drawback concerning the interpretation of the simulation results cannot be neglected. Further, especially the petri-nets are likely to restrict further refinements of the model especially concerning considerations on business objects exchanged.

Therefore, we decided to choose an agent-based simulation model as agents can more naturally model failure propagation in asynchronous environments. Further agents can be re-used, e.g. if the scenario is extended to adaptive business application relocation strategies to compensate load-peaks or system-failures.

5) Map information model to executable *simulation model:* The information model, containing all relevant elements for the simulation, e. g. business applications, interfaces, business objects, etc., has to be mapped to an executable simulation model of the application landscape.

Agent-based simulation models can be derived at different levels of granularity that range between two extremes. On the one hand the most detailed representations represent all active parts, e.g. business process and the services that perform individual activities, as individual agents. On the other side it is also possible to represent subsets of active elements, e.g. hardware components, by agents that manage the interactions of logically associated elements.

Following the requirements for the information model given above, the individual *business applications* lend themselves to be described by agent models. Agent instances include the local rules that decide when to engage in the execution of a business service. While several operations, e. g. internal computationgs in an application, are to be mimiced by agent internal processing,

can communication actions be represented by means of inter-agent communication.

6) Identify *simulation parameters:* The relevant simulation parameters, complementing the information model defined above, have to be identified.

In the context of the banking example, the parameters necessary for simulating the behavior especially center around the mapping of service requests on business application level to th degree of infrastructure service consumption caused thereby. Additionally, in-application processing times for internal computations make up an important parameter of the simulation model as well as the latency of inter-application communication may be. Finally, the frequency for each of the business processes to be supported can be considered a valuable parameter.

7) Determine / estimate simulation parameter *values:* For each of the simulation parameters identified before, an value has to be determined. These values can be derived e. g. from estimations from the expertise of business application owners or to be more objective from empirical data on current behavior of the actual systems.

The *frequency of executions per business process* could be determined using a business perspective, e. g. by counting the number of money transfers per day. *Resource consumption* and are not that easy to determine, therefore a simple estimation can be used, assuming that an internal computation utilizes 100% of a CPU core for a certain roughly estimated timespan. *Communication latency* can be determined using stastic methods on the data contained in log-files of a small-scale experimentation enviromnent. Concerning the values for the simulation parameters, assumptions made have to be explicated, in order to facilitate future validation or adaption of the simulation model.

8) Create *evolution scenarios* of the landscape: Based on the status quo of the application landscape (current landscape) different evolution scenarios (planned landscapes) have to be created, which represent possible evolutions scenarios. These scenarios can then be compared based on the objectives in order to support the decision process concerning the future evolution of the application landscape via the simulation results.

Potential ways to distribute the service providing business applications to different hardware clusters are made explicit in the evolution scenarios. Thereby, the scenarios are developed by enterprise architects of the bank, taking into consideration similarities between the applications to distribute. Further scenarios are created by combinatorial search algorithms, finding pareto-optimal solutions in the distinct business processes. Optimal scenarios derived by the latter way are subsequently reviewed by architects, as they could be optimized according to performance aspects, but may be senseless in respect to business usage. Potential scenarios conforming to the simple example from Figure 3 are illustrated in the software maps shown in Figure 5.

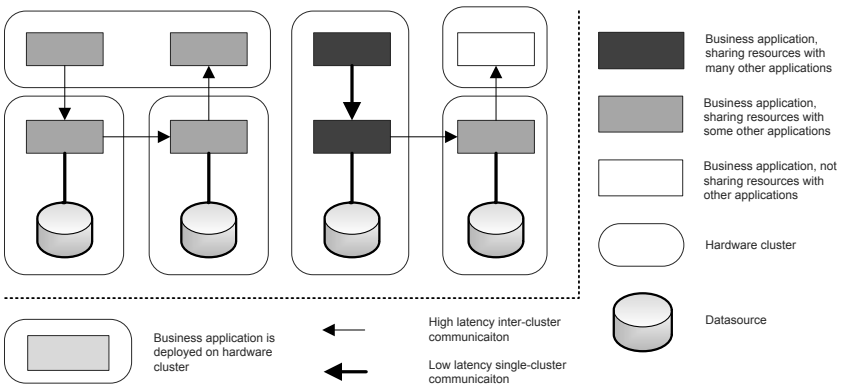

Figure 5: Distribution scenarios for the landscape involved regarding the execution of a business process

The evolution scenario, which achieved the higher outcome is implemented. Thereby the planned landscape is transformed to the current landscape.

9) *Validate* the simulation results with actual data: Based on the behavior of the new systems representing the previous planned landscape, the simulation results are validated by comparison with the actually measured empirical data of the new system.

> A new landscape has been implemented by projects and is finally operated by the IT staff. Performance measures regarding latency can be assessed directly by monitoring request processing time or indirectly by user satisfaction analyses. Especially direct results may be used to adapt simulation parameters, to achieve a better fitting. Indirect observations may at least be helpful to proof or neglect the simulation to be strongly deviating from reality.

In order to improve the simulation process introduced above, the findings of the final step *Validating the simulation results with actual data* should be used as input for the next iteration of the process. Thereby, each step, e. g. *the development of the information model* should be reconsidered in order to improve the outcomes of the simulation process, e. g. by introducing new concepts relevant for the simulation.

5 Critical reflection and outlook

In this paper we have motivated the necessity to enrich today's static EA models that focus on the static complexity arising from dependencies between the elements of the enterprise architecture by appropriate abstractions to capture the dynamic complexity of the EA understood as a socio-technical system of interacting systems. Based on related work on application landscape management, metrics, and simulation techniques established in

other management areas, we outlined an approach to EA simulations. We argued that simulations may provide decision support to develop better performing, more robust and adaptive EAs.

Nevertheless, in this paper we have neither provided a detailed information model nor a full-scale simulation model. To apply the approach presented, both models would be necessary but also dependent on the actual environment, in which simulation supported EA management was applied. Thereby, the steps 6)(identify simulation parameters) and 7)(determine / estimate simulation parameter values) of our approach form a major challenge in a practical environment. To address these issues a validation in the real world would be required. The real world example presented alongside the approach provides a real world example Although a real world example originating from practice is used to detail on the different steps of our simulation approach, case studies have to be accomplished in order to validate the presented approach in a practical setting. We argued that architectural decisions can not only be guided by metrics [Lan08, LS08] but may also be supported by predictive simulations of architecture designs. These designs provide so-called information models of the constituent architecture elements which are the subjects of simulation models. We discussed established simulation techniques and concluded that agent-orientation facilitates the derivation of simulation models. Furthermore agent technology provides a variety of simulation environments and tool support ready for use in the simulation methodology presented here.

Future prospects comprise the integration of application landscape modeling and simulation in an integrated tool chain. System dynamics research established *what-if* simulations as predictive tools to support management decisions (e.g. [Ste00]). Here we propose that EA architects would benefit from a similar approach. Crucial is the seamless integration of modeling and simulation tool support, i. e. being able to automate the derivation of simulation models from landscape descriptions. Since EAs depend on recurring types of architectural elements, companies may maintain a library of reusable simulations elements. These describe common architectural elements parameterized by specific information models. The here advocated agent–based simulation approach supports the required composition of simulation models. Since the complexity of these simulation models grows with the scale of the abstracted EA's, sophisticated simulation environments will be utilized [IGN05]. Particularly challenging in this context is ensuring the proper behavior of the system model, i. e. validating that the derived simulation elements behave as the architectural elements they represent. As this demands manual modeling and simulation effort, it is of interest to enable the reuse of simulation models, e. g. as executable architectural pattern.

References

[AS08] J.S. Addiks and U. Steffens. Supporting Landscape Dependent Evaluation of Enterprise Applications. In *Multikonferenz Wirtschaftsinformatik (MKWI) 2008*, pages 1815 – 1825, Munich, Germany, 2008.

[BELM08] S. Buckl, A. Ernst, J. Lankes, and F. Matthes. *Enterprise Architecture Management*

Pattern Catalog (Version 1.0, February 2008). Technical report, Chair for Informatics 19, Technische Universität München, Munich, Germany, 2008.

[BF04] Andrei Borshev and Alexei Filippov. From System Dynamics and Discrete Event to Practical Agent Based Modeling: Reasong, Techniques, Tools. In *Proceedings of the 22nd International Conference of the System Dynamics Society*, Oxford, UK, 2004.

[BLL+94] R. Bhaskar, H.S. Lee, A. Levas, R. Petrakian, F. Tsai, and B. Tulskie. Analyzing and re-engineering business processes using simulation. In *Proceedings of the 1994 Winter Simulation Conference*, pages 1206–1213, Lake Buena Vista, FL, USA, 1994.

[Der06] G. Dern. *Management von IT-Architekturen (Edition CIO)*. Vieweg, Wiesbaden, 2006. ISBN 528158166, (in German).

[EHH+08] G. Engels, A. Hess, B. Humm, O. Juwig, M. Lohmann, J.-P. Richter, M. Voss, and J. Willkomm. *Quasar Enterprise – Anwendungslandschaften serviceorientiert gestalten*. dpunkt.verlag, Heidelberg, 2008. ISBN 3898645061, (in German).

[FAW07] R. Fischer, S. Aier, and R. Winter. A Federated Approach to Enterprise Architecture Model Maintenance. In *Enterprise Modelling and Information Systems Architectures Proceedings of the 2nd International Workshop EMISA 2007*, pages 9–22, St. Goar, Rhine, Germany, 2007.

[Fok00] Wan Fokkink. *Introduction to Process Algebra*. Texts in Theoretical Computer Science. An EATCS Series. Springer, 2000. ISBN: 3-540-66579-X.

[For61] J.W. Forrest. *Industrial Dynamics*. MIT Press, 1961.

[Fra02] U. Frank. Multi-Perspective Enterprise Modeling (MEMO) – Conceptual Framework and Modeling Languages. In *Proceedings of the 35th Annual Hawaii International Conference on System Sciences 35*, Honolulu, 2002.

[Gro05] *The Open Group.* TOGAF "Enterprise Edition" Version 8.1., 2005.

[IEE00] IEEE. *IEEE Std 1471-2000: Recommended Practice for Architectural Description of Software-Intensive Systems*, 2000.

[IGN05] Toru Ishida, Les Gasser, and Hideyuki Nakashima, editors. *Massively Multi-Agent Systems I*, volume 3446 of *LNCS*. Springer, 2005.

[Jen96] K. Jensen. *Coloured Petri nets: basic concepts, analysis methods and practical use: volume 1*. Springer-Verlag, London, UK, 2nd edition, 1996. ISBN 3-540-60943-1.

[JVN06] M. Jansen-Vullers and M. Netjes. Business process simulation – a tool survey. In *Seventh Workshop and Tutorial on Practical Use of Coloured Petri Nets and the CPN Tools*, Denmark, 2006.

[KB03] S. Kounev and A. Buchmann. Performance Modelling of Distributed E-Business Applications using Queuing Petri Nets. In *Proc. of the 2003 IEEE International Symposium on Performance Analysis of Systems and Software*, pages 143–155, Austin, Texas, 2003.

[Kel07] W. Keller. *IT-Unternehmensarchitektur*. dpunkt.verlag, 2007. ISBN 3898644197 (in German).

[Lan05] M. Lankhorst. *Enterprise Architecture at Work*. Springer, Berlin, Heidelberg, 2005. 3540243712.

[Lan08] J. Lankes. *Metrics for Application Landscapes – Status Quo, Development, and a Case Study*. PhD thesis, Technische Universität München, Fakultät für Informatik, Munich, Germany, 2008. (submitted).

[LM04] M. Laguna and J. Marklund. *Business Process Modeling, Simulation, and Design*. Prentice Hall, 2004.

[LMW05] J. Lankes, M. Matthes, and A. Wittenburg. Softwarekartographie: Systematische Darstellung von Anwendungslandschaften. In *Wirtschaftsinformatik 2005*, Bamberg, Germany, 2005. (in German).

[LS07] J. Lankes and C.M. Schweda. *Constructing Application Landscape Metrics: Why & How*. Technical Report TB0701, Technische Universität München, Institut für Informatik, Lehrstuhl für Informatik 19, Munich, Germany, 2007.

[LS08] J. Lankes and C.M. Schweda. Using Metrics to Evaluate Failure Propagation in Application Landscapes. In *Multikonferenz Wirtschaftsinformatik (MKWI) 2008*, pages 1827 – 1838, Munich, Germany, 2008.

[LW04] K. Langenberg and A. Wegmann. *Enterprise Architecture: What Aspects is Current Research Targeting*. Technical Report C/2004/77, Ecole Polytechnique Fédérale de Lausanne (EPFL), France, 2004.

[MBLS08] F. Matthes, S. Buckl, J. Leitel, and C.M. Schweda. *Enterprise Architecture Management Tool Survey 2008*. TU München, Chair for Informatics 19 (sebis), Munich, Germany, 2008. ISBN 3-00-024520-0.

[Mog05] Jeffrey C. Mogul. *Emergent (Mis)behavior vs. Complex Software Systems*. Technical Report HPL-2006-2, HP Laboratories Palo Alto, 2005.

[MY04] XinJun Mao and Eric Yu. Organizational and Social Concepts in Agent Oriented Software Engineering. In *Agent-Oriented Software Engineering V, 5th International Workshop, AOSE 2004, Revised Selected Papers*, volume 3382 of *LNCS*, pages 1–15, New York, USA, 2004. Springer.

[PSR98] H. V. D. Parunak, Robert Savit, and Rick L. Riolo. Agent-Based Modeling vs. Equation-Based Modeling: A Case Study and Users' Guide. In *Proceedings of the First International Workshop on Multi-Agent Systems and Agent-Based Simulation*, pages 10–25, London, UK, 1998. Springer-Verlag.

[Ste00] John D. Sterman. *Business Dynamics – Systems Thinking an Modeling for a Complex World*. McGraw-Hill, 2000.

[SZ92] J. F. Sowa and J. A. Zachman. Extending and formalizing the framework for information system architecture. *IBM System Journal*, 31 (3), 1992.

[Tij03] H.C. Tijms. *A First Course in Stochastic Models*. Wiley & Sons, 2003.

[vdA03] W.M.P. van der Aalst. *Application and Theory of Petri Nets*, pages 1–22. Springer Verlag, Eindhoven, Netherlands, 2003.

[Wik08] Wikipedia. Wikipedia – Wikipedia, *The Free Encyclopedia*, 2008. [Online; accessed 22-July-2004], (in German).

[Wit07] A. Wittenburg. *Softwarekartographie: Modelle und Methoden zur systematischen Visualisierung von Anwendungslandschaften*. Phd thesis, Fakultät für Informatik, Technische Universität München, Munich, Germany, 2007. (in German).