An Application System Landscape Engineering Process Framework (ASLEP)

Completed Research

Johannes Hintsch
Otto von Guericke University Magdeburg
johannes.hintsch@ovgu.de

Karthik Gali
Otto von Guericke University Magdeburg
karthik.gali@ovgu.de

Naoum Jamous
Otto von Guericke University Magdeburg
naoum.jamous@ovgu.de

Klaus Turowski
Otto von Guericke University Magdeburg
klaus.turowski@ovgu.de

Abstract

The rapid growth of Information Technology (IT) usage in enterprises has led to complex, heterogeneous, and dynamic IT system landscapes. Thus, a realization of efficient and effective application system landscapes is required, which in turn demands for a synchronization of activities and collaboration among all involved stakeholders. In this research, we advocate an approach where implementation details can be shared with the customer from a birds-eyes viewpoint down to the technical implementation as needed. As the main artifact, a new framework named Application System Landscape Engineering Process (ASLEP) will be proposed. It supports the provider's service delivery process including cost estimation of application system landscapes. The research question addressed is: How can the ASLEP be formalized to integrate stakeholders, process stages, and information to build application system landscapes for customers of application service providers?

Keywords

Application Service provisioning, Application System Landscapes, Deployment, TOSCA.

Introduction

The realization of efficient and effective application system landscapes (ASL) requires proper synchronization among the activities and collaboration among the participating stakeholders. Application systems (Stahlknecht and Hasenkamp 2005, p. 226) consist of software for a particular application field and include programs and relevant data. Systems that support different application fields, and are integrated with other systems, form an application system landscape (Hintsch et al. 2016). In this paper, a broad definition of application systems, including virtualized or physical hardware, is used (Stahlknecht and Hasenkamp 2005, p. 226). Application service providers (ASP), design and operate ASLs for their customers (Riemer and Ahlemann 2001).

IT outsourcing (ITO) has been an approach to reduce costs spent on IT and to focus on core competencies for some decades (Lacity et al. 2010). However, the successful outcome of an IT outsourcing engagement does not come naturally. Effective knowledge sharing and communication are essential and are positively correlated with successful ITO engagements (Lacity et al. 2010). Information systems capability, both from a technical as well as methodological perspective, are essential aspects of outsourcing success. They are required both on the side of the provider, as well as that of the customer (Lacity et al. 2010).

Common practice frameworks (Hochstein et al. 2005) for IT service management such as the IT infrastructure library (ITIL) advocate hierarchical control of a provider's service portfolio (Cannon 2011). ITIL defines customer-facing services that are used by the customer to achieve a business outcome. Supporting services are used internally by the providers as a base for their service delivery to customers (Cannon 2011, p. 13). Teubner and Remfert (2017) state that this separation into customer-facing and supporting services is necessary to shield the customer from the complexities of implementation. However, while a product-based approach to IT service provisioning may be adequate for some customers (Zarnekow and Brenner 2003), considering only business value may not be adequate for all. Consider the case of an ASP that operates application system landscapes that are customized and extended by the customer. This customer will need to know technical details. Furthermore, in the discussions on DevOps (Alt et al. 2017) transparency is required. Recent data shows an increase in internal IT staff, internal IT user satisfaction as a prime performance indicator for IT, and integration as software development's most frequent activity (Kappelman et al. 2018). If outsourcers want to sell services to increasingly IT-aware clients, transparency of the sold services, especially on a technical layer, is important, particularly for integration projects building user-accepted services. Therefore, in this paper, we advocate an approach where providers can share implementation details with the customer from a birds-eyes viewpoint down to the technical implementation.

Also, we have learned in industry projects (such as with an ASP for telecommunications or a full IT outsourcer), that ASPs face difficulties in efficiently operating their *delivery pipeline*. This pipeline includes tasks from handling the order over designing the service to the deployment of the ASL. The sales department would, based on customer requirements, create an offer to the customer as an assembly of services using a service catalog. If the customer accepts the offer the sales department orders the application system landscape architect (ASLA) to design the system landscape. Finally, the ASLA sends the system landscape design to the application system landscape engineer (ASLE) who is tasked with the creation of the target system landscape.

Implementing the above-described process for the design and creation of ASLs is inefficient in terms of effort and communication. Critical further issues that were identified in our industry projects are:

1. Inaccurate estimation of costs, i.e. initial cost estimates offered to the customer are in most cases either overestimated or underestimated.
2. Inability to deliver the committed services due to technically infeasible solutions which are not taken into consideration while making the offer to the customer.

Within this paper the following research question (RQ) is addressed: *How can the Application System Landscape Engineering Process (ASLEP) be formalized to integrate stakeholders, process stages, and information to build an ASLs for ASPs customers?*

Following the design science research (Peffers et al. 2008), this paper is structured as follows. In the next section the research background will be described. Subsequently, related work will be discussed, and the research scope and gap highlighted. The ASLEP framework will be presented in detail in the third section, starting with its requirements. The evaluation of the presented artifact is the content of section four, and section five concludes the paper.

Research Background

In order to balance agility and consistency in aligning business requirements with IT implementations, enterprise architecture (EA) is widely accepted as an essential mechanism (Winter and Fischer 2007). Lankhorst (2004) differentiates three layers of EA. The *business* layer offers products and services to external customers. These products and services are realized in the organization by business processes performed by business actors. The *application* layer supports business layer with the application services, realized by application software. The *technology* layer provides infrastructural services (e.g. processing, storage and communication services) needed to run application software. It is realized by computer and communications hardware as well as system software.

According to Ruffin et al. (2001), "[...] the services offered to a customer by the service provider may take the form of enterprise business analysis (EBA), network computing, enterprise resource planning (ERP) application provision, electronic business (e-business) applications provision or any other pertinent IT solution services which may be of interest to a customer." Services addressed in this paper are application services provided by ASLs for customers on the basis of enterprise application software such as ERP, customer relationship management (CRM) or business intelligence (BI).

Übernickel et al. (2006) observed a shift within IT departments that focused on delivering IT functionality with catalog-based IT services instead of individual IT projects. To support this new paradigm, they present a reference process for developing IT services. It includes the steps of performing a customer requirements analysis, mapping of business functions to IT services, bundling of supporting services to customer-oriented services, performing a customer acceptance test, and finally cataloguing the service. Although a meta-model for describing IT services is conceptually outlined, no illustration of how instantiations of this model can be used in the respective steps is provided. A meta-model for IT services is also provided by Frank et al. (2009). With their IT domain specific modeling language (ITML), they focus on providing graphical modeling to users with different technical knowledge to achieve a better business / IT alignment. As such, it falls short of offering hints as to how operationalizable specifications can be derived from the models. Table 1 compares three approaches towards modeling and operationalizing ASLs for potential employment in ASLEP.

Framework	**UML**	**ITML**	**TOSCA**
Reference	Oestereich and Scheithauer (2012)	Frank et al. (2009)	OASIS (2015)
ASL modeling	Yes	Yes	Yes
Landscape meta-model	Generic	IT specific	Generic
Operationalizable ASL	Extension required	Extension required	Yes
Recognized standard	Yes	No	Yes
Cloud ecosystem support	*n.a.*	No	Increasing

Table 1 Three alternatives for modeling and deployment of ASLs in ASLEP.

The unified modeling language (UML) provides various diagram types that can be used towards modeling application system landscapes. Component diagrams may document architectures of service-oriented and component-based architectures. Deployment diagrams provide information about the allocation of application software on virtual and physical machines (Oestereich and Scheithauer 2012, p. 331). Message-based interaction of systems or components can be shown with communication diagrams (Oestereich and Scheithauer 2012, p. 356). The general-purpose language UML even has a specialization in the form of SySML for systems engineering (OMG 2015) that has a meta-model more specific to that domain. Using UML's profiling mechanism, which is based on stereotypes, Herden (2013) discusses how to transform graphical diagrams into configuration specifications that can be executed in order to deploy application system landscapes. However, it remains a scientific work without practical dissemination. The ITML focuses on documentation rather than creating operationalizable configuration specifications, although this would be possible as the meta-model could likely be transformed by using similar mechanisms as shown by Herden (2013). Amongst the three frameworks, the OASIS topology and orchestration specification for cloud applications (TOSCA) fulfills all comparison criteria (OASIS 2015). TOSCA has an underlying meta-model that consists of nodes and edges. These edges can represent arbitrary systems if the node and edge types are provided. Ecosystem support is increasing. For instance, an academic tool suite for modeling application system landscapes and for transforming them into deployable cloud service archive (CSAR) is available as open source[1]. The company CLOUDIFY[2] offers a public cloud service for deploying TOSCA-based ASLs. However, TOSCA makes no recommendation as to how an ASLEP could be structured. Substantial work in TOSCA has been performed from a computer science perspective (Wettinger et al. 2013, 2014). Information systems related research on TOSCA that is relevant to our research goal could not be identified.

The ASLEP Framework

In the first part of this section, ASLEP terminology and basic concepts are discussed based on a meta-model. Then, the Application System Landscape Engineering Process (ASLEP) framework is presented.

[1] http://www.opentosca.org/

[2] https://www.cloudify.co

Meta-model

The term entity here refers to the building blocks of the ASLEP framework. Five entities are differentiated. Figure 1 displays the entities and their relationships in an excerpt of the ASLEP's meta-model. For the meta-model, a notation ajar to UML class diagrams (Oestereich and Scheithauer 2012, p. 227) is used, omitting cardinalities.

Stakeholders refer to individual people or teams involved with the sales, design, and creation of ASLs. The stakeholders are broadly categorized into two categories as internal and external stakeholders. External stakeholders can be customers or suppliers. From the perspective of the ASP, everyone within the service provider's organization involved in delivering services to the customers are considered as internal stakeholders. A sales representative is the ASP's point of contact for the customers to convey their system landscape requirements. This role is responsible for the interaction with the customers to gather the ASL's requirements, create and forward the sales order as a delivery order to the ASLA. The ASLA is responsible for the requirement analysis, planning the delivery activities, and designing the architecture of the system landscapes. The application administrator is responsible for the deployment and maintenance of the application software components. The server administrator is responsible for the provisioning and maintenance of the computing and storage capacities part of system landscape. The network administrator is responsible for the provisioning of the network services and maintenance of the network appliances. Comparing the three types of administrators against the layers of Enterprise Architecture (EA), the business layer seems to be missing. However, administration of the technology layer typically involves networks and servers, which can be physical or virtual machines and which include for the server administrator operating systems (Schuff and St. Louis 2001). The ASLA acts as an interface to the business layer (sales representative and customer).

Various stages are passed through during the realization of an ASL. The stakeholders are associated with one or more of these stages. EA frameworks also organize the construction of the enterprise architecture (Group 2009). However, in this paper, EA's subphases and detailed activities involved with the development of EA's technology layer are discussed rather than the overall enterprise architecture.

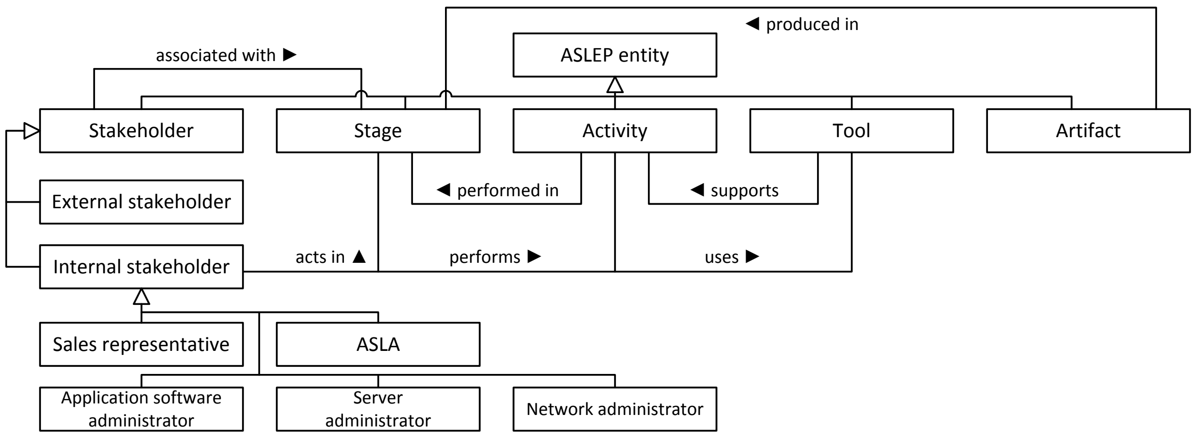


Figure 1 Excerpt of the ASLEP's meta-model.

Activities are the tasks performed by the stakeholders in each stage of developing the artifacts. An artifact is a piece of information in the form of text, model, or executable code. In the context of ASLEP, artifacts are classified into intermediate and final artifacts. The completion of activities within each stage produces an intermediate artifact. These intermediate artifacts mature in the subsequent phases. The fully matured final artifact is used to deploy the ASL in the final stage.

Tools are used by the stakeholders to perform their activities. For example, modeling tools such as UML can be used by the architect to design the conceptual model of the landscape. Text documents or CRM software could be used by the sales representative for gathering the customer requirements.

In the next subsection, the ASLEP framework is presented.

Application System Landscape Engineering Process (ASLEP)

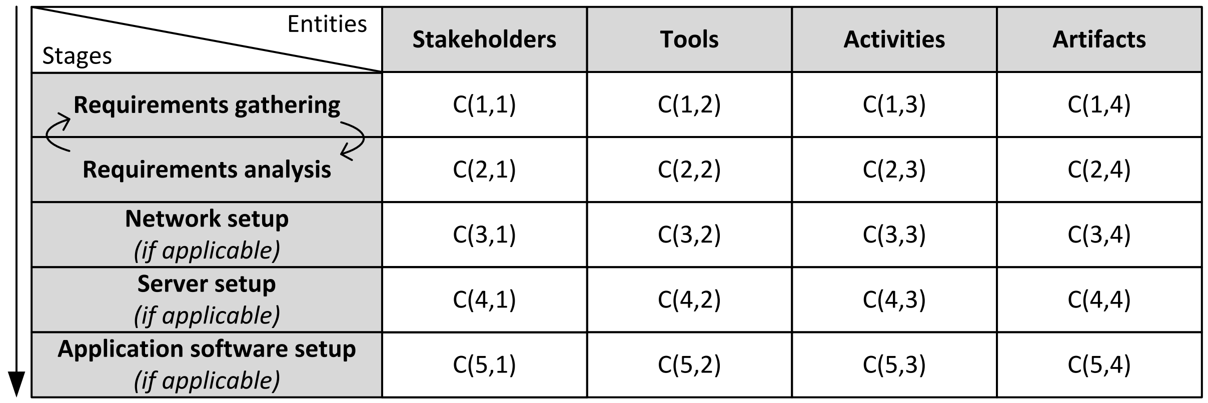
The objectives of the ASLEP address the issues raised in the introduction.

Req. 1) Provider's costs for production of an ASL can be estimated accurately at point of sales.

Req. 2) Committed services can be delivered.

The ASLEP framework is organized in a two-dimensional (5 x 4) matrix as it is shown in Figure 2. It shows for each stage (represented in rows) what stakeholders, tools, and activities are involved to create which artifacts in order to design and create ASLs. The cells of the matrix are referred as C(x, y), where 'C' stands for the cell, 'x' represents the sequential number of the stage and 'y' represents the sequential number of the ASLEP entity corresponding to the stage.

The outer arrow in Figure 2 shows the general order in which the stages are passed through. However, several iterations may be needed before the requirements are gathered in such a way that they can be fully analyzed. The output of the requirements analysis is an operationalizable model of the ASL that, at the same time, serves as a functional diagram for discussions with the customer. Each node of the diagram is associated with a deployment and an implementation artifact (Binz et al. 2014) that is used, for example, to deploy a configured web server. If all nodes' deployment and implementation artifacts and all relationship types are available then the network, server, and application software setup stages can be skipped. If only deployment and implementation artifacts of one stage, say server, need to be created then only this stage would be included in that particular ASLEP instance in addition to the first two stages.



Entities / Stages	Stakeholders	Tools	Activities	Artifacts
Requirements gathering	C(1,1)	C(1,2)	C(1,3)	C(1,4)
Requirements analysis	C(2,1)	C(2,2)	C(2,3)	C(2,4)
Network setup *(if applicable)*	C(3,1)	C(3,2)	C(3,3)	C(3,4)
Server setup *(if applicable)*	C(4,1)	C(4,2)	C(4,3)	C(4,4)
Application software setup *(if applicable)*	C(5,1)	C(5,2)	C(5,3)	C(5,4)

Figure 2 The ASLEP framwork.

Requirements Gathering

The ASLEP process begins with the requirement gathering stage. The sales representative has access to the full-service catalog offered by the ASP. The utility of application services is determined by both their functional and non-functional characteristics (Chung and Prado Leite 2009). Therefore, customer requirements are broadly classified into two categories:

1. Clarification of which application software components need to be hosted.
2. Clarification of non-functional requirements that need to be satisfied.

C (1,1) The primary stakeholders involved with the requirement gathering stage are sales representative and the customer. In cases of an internal IT service provider (Becker et al. 2011), the customer might also be from within the same organization as the ASP. **C (1,2)** The framework suggests the use of an order management system as found in ERP (Davenport 2000) or IT service management systems (Lloyd 2011, p. 145). The customer requirements are attached to an inquiry. **C (1,3)** The activities of the sales representative include the capturing of all customer requirements. They are recorded in the service level agreement (SLA). SLA's can be derived from standard components (Skene et al. 2010). If the customer agrees to the results of the requirements analysis, a sales order may be generated from the inquiry. Depending on the scenario, there might be some iterations before the customer is ready to sign a contract and a sales order is generated. **C (1,4)** The artifacts that are created in this stage are inquiry, sales order, SLA, and contract.

Requirement Analysis

C (2,1) The ASLA is responsible for designing the application system landscape on an abstract level. The ASLA has a crucial role because the architect needs to understand the technical feasibility of possible solutions. Of course, following the sharing principle of DevOps (Alt et al. 2017), network, server, and application software teams should be involved and readily assist when necessary. **C (2,2)** Based on the discussions in the research background, the TOSCA framework is suggested for modeling and operationalizing ASLs. Therefore, a TOSCA based modeling tool is used in this stage. **C (2,3)** From the ASP perspective, server-side components need to be planned for hosting the applications and their required server. The application software components that need to be hosted determine the rest of the landscape components. Therefore, the ASLA analyses the requirements and identifies all the landscape components required to deliver the target system landscape. Eventually, the ASLA has created an architecture description of the ASL. **C (2,4)** The intermediate artifact of the requirement analysis stage is the conceptual model of the ASL. Regarding TOSCA, this is an application topology model representing the ASL's components and their functional relationships (Binz et al. 2014).

Network Setup

C (3,1) The network administrator is the first administrator to become active. The networks need to be available for hosts to communicate, which is why this stage comes first. Non-functional requirements are tightly coupled with the landscape's network and server components. For example, the number of CPU cores determines the system performance or response times. The data transfer rates are dependent on the network bandwidth. The provisioning of network and server resources requires precise planning, as the over-provisioning of resources to avoid SLA penalties, results in inefficient utilization of the resources and higher costs of operation while under provisioning can result in penalties or customer dissatisfaction (Skene et al. 2010). **C (3,2)** Tools required by the network administrator include the TOSCA modeling tool and common tools such as editors and terminals. Also, a version control system is used to store intermediate artifacts. **C (3,3)** In the context of TOSCA, the network administrator needs to perform the following activities (Binz et al. 2014).

1. Equip each node template with a deployment artifact.
2. Define the management operations for each of these node templates. For example, the management operation of a firewall can be ENABLE or DISABLE, which can then be used by the implementation artifacts.
3. Create the implementation artifacts (e.g. scripts) for implementing the management operations of the node templates.
4. Define the properties and management operations for each of the relationship templates.
5. Create the implementation artifacts to implement the management operations of the relationship templates.
6. Define the management plans for all the network components.

C (3,4) The intermediate artifacts, which need to be created, were discussed in the discussion on the required activities. They are stored in the version control system.

Server Setup

C (4,1) The server administrator is responsible for the creation of the substantiating the topology in the same way the network administrator did (e.g., creating deployment artifacts, etc.). **C (4,2)** Similar tools, to those used by the network administrator, will also be used by the server administrator. **C (4,3)** The activities surround the creation of intermediate artifacts for the servers and operating systems. TOSCA is designed for cloud computing. Therefore, the approach in this paper is more streamlined if the servers are virtual machines. However, current IaaS platform software, for virtualization, also support the integration of physical (bare metal) server in their management scheme (Foundation 2011). **C (4,4)** The intermediate artifacts of this stage correspond to those of the network setup stage.

Application Software Setup

C (5,1) The application software administrator is responsible for the installation of all landscape software components. **C (5,2)** The tools correspond to those used by the other administrators, although product-specific additional tools may be necessary for administering application software. Customization and development are not in scope of this paper. If they were, corresponding software engineering tools would also be necessary. **C (5,3)** The activities also correspond to those of the other setup stages. **C (5,4)** Lastly, the intermediate artifacts correspond to those produced in the other setup stages.

At this stage, based on all intermediate artifacts, the final artifact can be generated. The CSAR includes all intermediate artifacts necessary for deploying the ASL for the customer.

The ASLEP framework, which was presented in this section, will be evaluated in the next section.

Evaluation

The evaluation of the ASLEP framework is conducted by employing a stage-wise evaluation method (Sonnenberg and Brocke 2012). For design science research it is often difficult to proof final validity. In particular, if field studies were not conducted critics often deny respective design science research a truth-like value. Therefore, Sonnenberg and Brocke (2012) propose that design science research should be evaluated in four stages: two that are conducted prior to instantiating the artifact (ex-ante) and two after it has been instantiated (ex-post). Within this paper, only the first three evaluation stages are passed through. Implementation and testing in a real-world case (EVAL 4) is still pending.

For EVAL1, the validation of the research design, we follow an argumentative approach. As we have followed the guidelines by Peffers et al. (2008) our research design is justified. In particular, we have stated a relevant problem. The research question addressed in this paper is:

How can the Application System Landscape Engineering Process (ASLEP) be formalized to integrate stakeholders, process stages, and information to build an ASLs for ASPs customers?

This question has not been sufficiently addressed by previous literature, which is why the research gap is justified. The research question has been answered in the form of the ASLEP framework. Based on experience with our industry partner two requirements were defined. As these requirements pertain to the necessities illustrated for the ASLEP scenario, we argue that they are justified.

In EVAL2, the design specification and its objectives are justified. This is first demonstrated by instantiating the ASLEP framework. This instantiation is shown in Figure 3. The Figure shows the different entities of the ASLEP framework in an overview and how, through the progression of stages, the artifacts are transformed from inquiry to the deployable CSAR.

Figure 3 also represents EVAL3. To verify that the ASLEP is technically implementable a simple example as a prototype: A customer inquiries about an ERP service which his company needs for testing a new business scenario related to industry 4.0. The inquiry states that the customer needs to be made a good offer. The order is consequently completed with a supporting service and component description of how the service should be constructed. In addition, the SLA is specified. The ASLA will then draw the bird-eye view on the application system landscape. It includes an SAP ERP installation with a preloaded master data from the Global Bike Inc. test company. The administrators will consequently add to the CSAR: the instructions regarding the network, server and operating system, and the Puppet[3] module for installing the ERP system together with its database installation that is run in our data center and an OpenTOSCA installation in a container that runs on a desktop machine. OpenTOSCA uses the CSAR to access the OpenStack installation in order to orchestrate the deployment of the service.

[3] Puppet (https://puppet.com/) is a configuration management software. For the prototype, we have used a setup including a server-based OpenStack (https://www.openstack.org/) installation for IaaS.

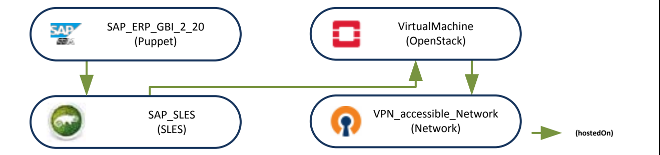
Entities / Stages	Stakeholders	Tools	Activities	Artifacts
Req. gathering	Sales representative and customer	Order management system	C(1,3)	Inquiry: *Customer requires ERP service for testing a new Industry 4.0 scenario (must be price-effective!!)* — trans-ferred → Order: Supporting servcies and components: *SAP ERP on MaxDB with GBI_2_20 import, on* SLES, xl-sized VM, 100GB network traffic volume, VPN-protected access point; SLA: *DEV usage, Office reaction times*
Req. analysis	ASLA	OpenTOSCA Modelling	C(2,3)	 SAP_ERP_GBI_2_20 (Puppet) VirtualMachine (OpenStack) SAP_SLES (SLES) VPN_accessible_Network (Network) (hostedOn)
Network setup	Network administrator	OpenTOSCA Modelling Admin. tools Version mgmt.	C(3,3)	yaml excerpt: `... VPN-access: type: tosca.nodes.nfv.CP properties: floatingIP: false ...` `... requirements: - virtualBinding: VDU1 - virtualLink: private ...`
Server setup	Server administrator	OpenTOSCA Modelling Admin. tools Version mgmt.	C(4,3)	yaml excerpt: `... SAP_SLES_XL: type: tosca.nodes.Compute capabilities: host: properties: num_cpus: 4` `... mem_size: 16384 MB os: properties: architecture: x86_64 type: Linux distribution: SLES`
Application software setup	Application software administrator	OpenTOSCA Modelling Admin. tools Version mgmt.	C(5,3)	puppet deployment: `class ac_saperp::sap_instal { exec { 'install_sap' : cwd => '/sapinst/installfiles/SPM/SPM_extracted', command => '/sapinst/installfiles/SPM/SPM_extracted/sapinst SAPINST_PARAMETER_CONTAINER_URL=/sapinst/customini/inifile.xml SAPINST_CWD=/sapinst SAPINST_EXECUTE_PRODUCT_ID=NW_ABAP_OneHost:NW702.ADA.CP SAPINST_SKIP_DIALOGS=true -nogui -noguiserver', [...] }}`

Figure 3 Instantion of the ASLEP prototype artifacts.

How the requirements of the ASLEP are addressed is discussed below.

Req. 1: Provider's costs for production of an ASL can be estimated accurately at point of sales. The costs of engineering an ASL can be accurately estimated at the beginning of customer engagement. During the requirements analysis stage, the requirements are transformed into a functional specification of the application system landscape. The functional requirements can be used as a basis for discussing with the customer if the landscape is designed as required. In addition, the architect can calculate an estimate based on the kinds of nodes he employs. Existing nodes templates are priced based on their procurement costs (e.g. for software licenses or monthly server operation costs) and the time spent by employees on developing them. Hence, the ASL topology can be inspected and costs can be summed up. If certain node templates do not exist, it would be possible to estimate their development costs base on similar node templates. For this, a detailed project management database or controlling system is crucial to identify the time spent on the similar node template.

Req. 2: Committed services can be delivered. The information regarding all layers of the application system landscape, including the information necessary to deploy the ASL, is available in the CSAR. Therefore, committed service should be deliverable in most cases. In particular, if node templates are created with constraints, only "allowed" compositions will be possible. Furthermore, in automated and virtualized environments, testing of the landscape can be done very fast. The same or a slightly modified CSAR can be used to deploy a test landscape in a quality assurance environment. This automated deployment reduces significantly the testing costs for a company employing an automated testing approach within their DevOps implementation (Alt et al. 2017).

The ASLEP framework attempts to identify and integrate the multiple elements involved with provisioning of the application services in the form of ASLEP framework entities. It describes the stakeholders involved with provisioning of the system landscapes, their responsibilities, the tools to perform their activities and the delivered artifacts for the communication with other stakeholders involved in the process. Also, the framework suggests the use of an order management system to handle the requests of customer sufficiently. Since all information is readily available in the order management system, the version control system, and finally in the CSAR, thus, communication will be more efficient. If specific node templates are reused, manual effort can be spared.

Conclusion

The current trend towards outsourcing IT services distances the customer organization from much of the direct costs incurred in IT maintenance and operations. As the IT service provider takes over the responsibility of meeting an organization's IT needs partly or wholly, Therefore, the provider should work more efficiently, communicate better with the customer, and be able to deliver the committed services.

To support the engineers, IT operation managers and project managers of an enterprise, this paper presented the application system landscape engineering process (ASLEP) framework. It aims to enable efficient communication among the involved stakeholders for the design and creation of ASLs. ASLEP integrates the sequence of all the activities, responsibilities of the involved stakeholders, artifacts produced by stakeholders, tools to develop the artifacts at each stage of the process to build the target ASL. In addition, the framework suggests the use of the Topology and Orchestration Specification for Cloud Applications (TOSCA) specification to automate the provisioning of ASLs in order to make the provisioning process efficient in effort and therefore time-to-market.

Future work should include evaluating the ASLEP's applicability for brownfield approaches. In such approaches, extension of existing system landscapes would be required. The operations and maintenance stages are not discussed in this paper as well as qualitative aspects of service delivery, which should be done in future work. However, the TOSCA specification has provisions for defining backup plans or for scaling services. This work did not consider the managerial aspects such as contracts, sign-offs, and financial matters in detail. These will be addressed to extend the framework in that direction.

REFERENCES

Alt, R., Auth, G., and Kögler, C. 2017. "Innovationsorientiertes IT-Management – Eine Fallstudie zur DevOps-Umsetzung bei T-Systems MMS," *HMD Praxis der Wirtschaftsinformatik* (54:2), pp. 216–229.

Becker, J., Poeppelbuss, J., Venker, D., and Schwarze, L. 2011. "Industrialisierung von IT-Dienstleistungen: Anwendung industrieller Konzepte und deren Auswirkungen aus Sicht von IT-Dienstleistern," in *Tagungsband der 10. Internationale Tagung Wirtschaftsinformatik* (*WI*), A. Bernstein and G. Schwalbe (eds.), Zurich, Switzerland: AIS Electronic Library, pp. 345–354.

Binz, T., Breitenbücher, U., Kopp, O., and Leymann, F. 2014. "TOSCA: portable automated deployment and management of cloud applications," in *Advanced Web Services*, New York: Springer, pp. 527–549.

Cannon, D. 2011. *ITIL – Service Strategy*, Norwich: The Stationery Office.

Chung, L., and Prado Leite, J. C. S. do. 2009. "On non-functional requirements in software engineering," in *Conceptual modeling: Foundations and applications*, Berlin Heidelberg: Springer, pp. 363–379.

Davenport, T. H. 2000. *Mission Critical: Realizing the Promise of Enterprise Systems*, Boston, USA: Harvard Business School Press.

Foundation, O. 2011. "Hypervisor Support Matrix," (available at https://wiki.openstack.org/wiki/HypervisorSupportMatrix).

Frank, U., A, D. F. F., Heise, D., B, E. H., Kattenstroth, H., and C, M. G. W. 2009. "ITML: A Domain-Specific Modeling Language for Supporting Business Driven IT Management," in *Proceedings of the 9th OOPSLA workshop on domain-specific modeling* (*DSM*), M. Rossi, J. Sprinkle, J. Gray, and J.-P. Tolvanen (eds.), Orlando, USA: HSE Print, pp. 4:1–4:8.

Group, T. O. 2009. *The Open Group Architecture Framework: Version 9*, The Open Group.

Herden, S. 2013. *Model-Driven-Configuration-Management: Ein modellgetriebener Ansatz für das Konfigurationsmanagement von IT-Systemlandschaften*, Wiesbaden: Springer Vieweg.

Hintsch, J., Kramer, F., Jamous, N., and Turowski, K. 2016. "The Application System Landscapes of IT Service Providers: A Multi Case Study," in *Proceedings of the 4th International Conference on Enterprise Systems* (*ES*), G. Li and Y. Yu (eds.), Los Alamitos, California, Washington, Tokyo: IEEE, pp. 122–131.

Hochstein, A., Zarnekow, R., and Brenner, W. 2005. "ITIL as common practice reference model for IT service management: formal assessment and implications for practice," in *International*

Conference on e-Technology, e-Commerce and e-Service (*EEE*), W. K. Cheung and J. Hsu (eds.), Hong Kong, China: IEEE, pp. 704–710.

Kappelman, L., Johnson, V., Maurer, C., McLean, E., Torres, R., David, A., and Nguyen, Q. 2018. "The 2017 SIM IT Issues and Trends Study," *MIS Quarterly Executive* (17:1), pp. 53–88.

Lacity, M. C., Khan, S., Yan, A., and Willcocks, L. P. 2010. "A review of the IT outsourcing empirical literature and future research directions," *Journal of Information Technology* (25:4), pp. 395–433.

Lankhorst, M. 2004. "Enterprise architecture modelling – the issue of integration," *Journal of Advanced Engineering Informatics* (18:4).

Lloyd, V. 2011. *ITIL – Continual Service Improvement*, Norwich, UK: The Stationery Office.

OASIS. 2015. "TOSCA Simple Profile in YAML Version 1.0," (available at http://docs.oasis-open.org/tosca/TOSCA-Simple-Profile-YAML/v1.0/TOSCA-Simple-Profile-YAML-v1.0.doc).

Oestereich, B., and Scheithauer, A. 2012. *Analyse und Design mit der UML 2.5: Objektorientierte Softwareentwicklung*, München: Oldenbourg Wissenschaftsverlag.

OMG. 2015. *OMG Systems Modeling Language, Version 1.4*.

Peffers, K., Tuunanen, T., Rothenberger, M. A., and Chatterjee, S. 2008. "A Design Science Research Methodology for Information Systems Research," *Journal of Management Information Systems* (24:3), ME Sharpe, pp. 45–77.

Riemer, K., and Ahlemann, F. 2001. "Application Service Providing – Erfahrungsbericht aus Sicht eines Providers," in *Tagungsband der 5. Internationale Tagung Wirtschaftsinformatik*H. U. Buhl, A. Huther, and B. Reitwiesner (eds.), Augsburg, Germany, pp. 743–756.

Ruffin, M., Jayaram, K. R., Merenda, A. C., Morrison, T. I., Ordonez, C. A., Preston, A. H., Temple, J. L., and Yan, E. L. 2001. "Method, system and program product for evaluating the business requirements of an enterprise for generating business solution deliverables,".

Schuff, D., and St. Louis, R. 2001. "Centralization vs. Decentralization of Application Software," *Commun. ACM* (44:6), New York, NY, USA: ACM, pp. 88–94.

Skene, J., Raimondi, F., and Emmerich, W. 2010. "Service-Level Agreements for Electronic Services," *IEEE Transactions on Software Engineering* (36:2), pp. 288–304.

Sonnenberg, C., and Brocke, J. vom. 2012. "Evaluations in the Science of the Artificial – Reconsidering the Build-Evaluate Pattern in Design Science Research," in *Design Science Research in Information Systems. Advances in Theory and Practice: 7th International Conference, DESRIST 2012, Las Vegas, NV, USA, May 14-15, 2012. Proceedings*K. Peffers, M. Rothenberger, and B. Kuechler (eds.), Berlin, Heidelberg: Springer Berlin Heidelberg, pp. 381–397.

Stahlknecht, P., and Hasenkamp, U. 2005. *Einführung in die Wirtschafsinformatik* (11th ed.), Berlin et al.: Springer.

Teubner, A., and Remfert, C. 2017. "Giving IT Services a Theoretical Backing," in *Human Interface and the Management of Information: Information, Knowledge and Interaction Design: 19th International Conference, HCI International 2017, Vancouver, BC, Canada, July 9–14, 2017, Proceedings, Part I*S. Yamamoto (ed.), Cham: Springer International Publishing, pp. 448–468.

Übernickel, F., Bravo-Sánchez, C., Zarnekow, R., and Brenner, W. 2006. "IS Service-Engineering: A process model for the development of IS services," in *European and Mediterranean Conference on Information Systems* (*EMCIS*), Z. Irani, O. D. Sarikas, J. Llopis, R. Gonzalez, and J. Gasco (eds.), Costa Blanca, Alicante, Spain, p. C29.

Wettinger, J., Behrendt, M., Binz, T., Breitenbücher, U., Breiter, G., Leymann, F., Moser, S., Schwertle, I., and Spatzier, T. 2013. "Integrating Configuration Management with Model-driven Cloud Management based on TOSCA," in *3rd International Conference on Cloud Computing and Service Science* (*CLOSER*), SciTePress, pp. 437–446.

Wettinger, J., Breitenbcher, U., and Leymann, F. 2014. "Compensation-Based vs. Convergent Deployment Automation for Services Operated in the Cloud," in *Service-Oriented Computing* (*Lecture Notes in Computer Science*), X. Franch, A. Ghose, G. Lewis, and S. Bhiri (eds.) (Vol. 8831), Springer, pp. 336–350.

Winter, R., and Fischer, R. 2007. "Essential Layers, Artifacts, and Dependencies of Enterprise Architecture," *Journal of Enterprise Architecture* (3:2).

Zarnekow, R., and Brenner, W. 2003. "A Product-Based Information Management Approach," in *Proceedigns of the 11th European Conference on Information Systems* (*ECIS*), R. Claudio U.and Mercurio Ciborra, M. de Marco, M. Martinez, and A. Carignani (eds.), Naples, Italy: AIS Electronic Library, pp. 2251–2263.