3-4-2015

Industrialization in Cloud Computing with Enterprise Systems: Order-to-Cash Automation for SaaS Products

Johannes Hintsch

Holger Schrödl

Hans-Jürgen Scheruhn

Klaus Turowski

Industrialization in Cloud Computing with Enterprise Systems: Order-to-Cash Automation for SaaS Products

Johannes Hintsch[1], Holger Schrödl[1], Hans-Jürgen Scheruhn[2], and Klaus Turowski[1]

[1] Otto von Guericke University, Faculty of Computer Science, Magdeburg, Germany
{johannes.hintsch,holger.schroedl,klaus.turowski}@ovgu.de
[2] Hochschule Harz, Faculty of Automation and Computer Science, Wernigerode, Germany
hscheruhn@hs-harz.de

Abstract. Industrialization has led to efficiency and effectiveness gains in manufacturing industries. Industrialization approaches such as standardization or sourcing are also key concepts in modern service companies. Therefore, current IS research studies how to transfer industrialization approaches to IT service production in order to increase efficiency and effectiveness. Automated and integrated value creation, as supported by ERP systems, have only been limitedly researched in the context of IT service production, although companies see this as an important area of improvement. In this paper, a model of how to automate and integrate an order-to-cash process for providers who offer SaaS products is proposed. The focus lies on the service provisioning process step and the model is evaluated with a prototype implementation. This research has implications for researchers and practitioners alike, as it proposes how to more effectively and efficiently deliver IT services, and uses established IS research to do so.

Keywords: Computer-Integrated Manufacturing, DevOps, Enterprise Resource Planning, Order-to-Cash, Software as a Service.

1 Introduction

Under the phrase “industrialization of IT” researchers and practitioners have been looking to concepts of physical goods manufacturing for new insights to improve efficiency and effectiveness of producing IT services. Becker et al. [1] study if and how industrialization approaches are applied by IT service providers. As examples of such approaches, they identified product and process standardization, modularization, automatic and integrated value creation, sourcing (concentration on core competencies), and quality management (continuous improvement). According to their study, widely adopted approaches are process standardization, modularization, and quality management. The studied providers see further potential for improvement by adopting the approaches of sourcing, process standardization, modularization, and automated and integrated value creation. Sourcing, process standardization and modularization have been widely researched (e.g. [2–4]). Enterprise resource planning (ERP) systems, incorporating concepts such as computer-integrated manufacturing (CIM), are used to automate and integrate value creating business processes [5, 6]. However,

12th International Conference on Wirtschaftsinformatik,
March 4-6 2015, Osnabrück, Germany

Hintsch, J.; Schrödl, H.; Scheruhn, H.; Turowski, K. (2015): Industrialization in Cloud Computing with Enterprise Systems: Order-to-Cash Automation for SaaS Products, in: Thomas. O.; Teuteberg, F. (Hrsg.): Proceedings der 12. Internationalen Tagung Wirtschaftsinformatik (WI 2015), Osnabrück, S. 61-75

ERP systems' applicability in the IT service industry has only been limitedly researched [7], although physical goods manufacturing concepts have been shown to be transferable [8].
In this paper, we study how to improve automation and integration of value creation at IT service providers who utilize ERP systems. The integration of all systems on the same data basis in physical goods manufacturing is known as Product Lifecycle Management (PLM) [9] and, in the automobile industry, has a positive influence on corporate success [10]. In physical goods manufacturing, make-to-order companies, which make and assemble components, are differentiated from build-to-order companies, which only assemble existing parts [11]. The implementation of an ERP system is expensive in make-to-order companies due to their individuality, as pointed out by Stevenson et al. [12].
In this study we focus on SaaS (Software as a Service) providers, as their products are continuously gaining industry acceptance [13]. We anticipate that for these providers, the current technological state of the art offers standardization and modularization capabilities, that, combined with ERP and related CIM concepts, are sufficient to automate and integrate the value creation of SaaS providers who have a build-to-order production process.

1.1 Methodology

We employ the design science paradigm as outlined by Hevner et al. [14], and the research process is conducted as defined by Peffers et al. [15]. The process includes six steps: identify problem & motivate (1), define objectives of a solution (2), design & development (3), demonstration (4), evaluation (5), and communication (6).
The research problem was identified based on current scientific literature, and the research follows a problem-centered approach (1). The solution to the problem should improve efficiency and effectiveness of IT service production. In this research, human resources are not considered. There are tasks in SaaS production performed by human resources that we believe cannot be fully automated. Therefore, human resources are ignored in order to focus on the pure application side of SaaS production. The paper's focus lies on the provisioning of a service, and neither previous stages such as design, nor subsequent stages such as operation; in other words, the last step in the Information Technology Infrastructure Library's (ITIL) transition stage (2). The designed artifact is a process model of an order-to-cash process, which in the SaaS provisioning step features models detailing the application and infrastructure layer (3). The research is relevant to both researchers and practitioners. For the former because established theory is used to solve problems in a new domain, and for the latter because a generic way of automating and integrating a SaaS order-to-cash process is presented. The feasibility of the artifact is demonstrated with a prototype [15] (4), which is evaluated based on its capability to fulfill the outlined goals (5). The research results are communicated in this paper (6).

1.2 Structure of the Paper

In the next section the research background is presented. It is comprised of an overview of existing research on ERP systems for the IT service industry, similarities between computer-integrated manufacturing and the IT trend DevOps, as well as configuration management, and software and infrastructure as a service. On that basis, the identified research gap is described in subsection 2.4, which we try to close with the proposed solution described in the fourth section. First, a generic order-to-cash process is presented. Secondly, the analogy between automated and integrated physical goods production and SaaS production is described. Next, an extended SaaS model is presented, which details how SaaS resources can automatically be configured, and how the service model can be connected to the concept of bill of materials. Lastly, the interaction of the proposed components in the provisioning step of the order-to-cash process is described. The proposed solution is verified by a prototype implementation in the fifth section. The paper closes with a discussion and an outlook on future work.

2 Research Background

2.1 ERP Systems for the IT Service Industry

ERP systems exist for a wide range of industries. We focus on the IT service industry, and in this paper specifically on providers offering SaaS. Mell and Grance [16] define SaaS as one of three cloud computing service models, the others being Platform as a Service (PaaS) and Infrastructure as a Service (IaaS). Furthermore, Zarnekow and Brenner [17] define an IT product. A product is a service that satisfies a specific need and results in a specific benefit. They define an IT product on four different levels: level 1 as an IT resource such as computing power, level 2 as an IT solution such as an application, level 3 as IT process support such as IT support for a specific business process, and level 4 as a business product such as electronic tickets or internet access. The complexity and business orientation of an IT product increases with each level. A contract exists between the customer and the provider of an IT product, and it may specify different IT services [18]. Becker et al. [1] differentiate between internal and external providers and between providers with industrial and personnel value creation. While the latter relies mostly on human skills found in professional services, the former can be characterized as equipment based with a strong adherence to the approaches of industrialization, mentioned in the introduction. The uno-acto principle applies to IT services, which means that value creation and consumption happen simultaneously [19]. When speaking of IT service production, an IT service is in production when it was provisioned and is used by the user.
The ERP system research field has reached a certain maturity [6]. However, only limited research exists on ERP systems for the IT service industry [7]. Several German dissertations on the related topic production planning and control (PPS) have been published (e.g. [20–22]). Ebert et al. [18] developed a service model based on an

older model by Übernickel et al. [23]. They state it may be the initial step towards an ERP system for the IT service industry.
Pilgram and Vogedes [24] report that, according to their analysis, 80 % of the identified requirements for IT service providers are covered by the ERP system SAP Business Suite. However, they leave out how automated and integrated provisioning of IT products could be implemented. Pinnow [22] focuses on capacity planning in virtualized data centers. He shows how to use bin packing for the determination of the maximum number of virtual machines running on one host with acceptable performance.
In terms of service design, Böhmann et al. [2] describe how IT services may be modularized, and Übernickel et al. [23] describe a process model for IT service engineering. Processes are central to ERP systems and as such, process frameworks such as ITIL should be considered when studying ERP in this domain. For instance, Valiente et al. [25] propose an ITIL ontology which can be used to automatically find flaws in IT service provider's IT service management processes.
To the best of our knowledge, current literature lacks concepts of automated and integrated service provisioning in ERP-supported processes. For example, Ebert and Brenner [26] describe a high proportion of human labor in the provisioning of services such as hosting ERP systems.

2.2 Computer-Integrated Manufacturing and DevOps

In the manufacturing of physical goods things look different, particularly in the automobile industry. At the end of the last century, Scheer [27] proposed the Y-CIM model. It categorizes and relates the different information systems that take part in computer-integrated manufacturing. Parts can be drafted with computer aided design (CAD) models, their qualities evaluated with simulation tools, and the parts physically created by CNC-machines [9]. Manufacturing execution systems regulate the detailed production planning in factories on a daily basis and they are interfaced with ERP systems [9]. A tight integration of tools along the process chain requires a high degree of standardization, which is perceived by some as to debilitate creativity [10]. However, the utility of software support may be maximized when tools across the whole process and supply chain operate in an integrated way and on the same pool of data [9]. This is termed Product Lifecycle Management, and it entails that development and manufacturing departments have to work in close cooperation.
With software use becoming ubiquitous, demands have risen. New and legacy software artifacts have to be operated, and new software has to be developed and carefully fitted into existing production environments. The ITIL describes processes of how to deal with changes in the production environment in a structured way, as well as proposing databases to enhance consistency across the IT provider's operations. To the same effect, Spinellis [28] describes the young trend of DevOps. Because today's software systems have many dependencies, development and operation teams have to work together closely by such means as continuous deployment and automated testing. According to Spinellis, DevOps is particularly applicable to SaaS products or to customized applications such as SAP ERP [28]. Today, DevOps is also practiced by firms such as Facebook [29] and Google [30].

2.3 Configuration Management, SaaS, and IaaS

A major enabler of DevOps are configuration management tools [28]. These tools, available on a range of operating systems and architectures, take a model of a system's configuration. A model is stored in a repository in the form of a script. The tools translate models to device and operating system specific configurations [31]. A configuration management agent configures the device accordingly. Popular open source tools include CFEngine, Puppet, and Chef [28]. These tools allow the configuration of SaaS and PaaS offerings. An exemplary research study in this field aims at the direct integration of lower-level configuration models with high-level service models, at making the service runtime environment transparent for service model users, and at achieving portability of service models between cloud providers [32].
A popular example of Software as a Service is the customer relationship management offering by Salesforce.com. SaaS and PaaS can be provisioned on IaaS platforms. Different public IaaS offerings such as Amazon's EC2 or Rackspace exist. While Amazon uses its own software, RackSpace advertises its usage of OpenStack [33]. OpenStack is described by Forrester's analysts [34] as the new de facto model, which is also used in academia [35]. OpenStack may be deployed in a private, public, or hybrid cloud setting [36], and it can manage computing resources offered by virtualization hypervisors, para virtualized containers, or bare metal nodes [37].

2.4 Research Goal based on Research Gap

Trying to use production theory and related concepts such as computer-integrated manufacturing as an analogy for enhancing efficiency and effectiveness of IT service providers is not new. However, existing service models and approaches have not sufficiently covered the automated and integrated value creation. In particular, a detailed description of how the service provisioning in an order-to-cash process is to be integrated with an ERP system is missing. Looking at the automobile industry and their use of CIM, a transfer to the IT service industry is promising. In build-to-order companies, the sales department collects orders, and if capacity permits, the ordered product is made. NC-, CNC-, DNC-machines ((C/D)NC-machines) are used to build parts, and these are assembled in different work centers in a factory and the final product is shipped. Exactly how to transfer this production process to IT service provisioning has, to the best of our knowledge, not yet been covered. IaaS offerings and configuration management systems are generally applicable to SaaS deployments, which is why the model could be generalizable. Deep technical knowledge such as configuring landscape's systems can be formalized, and configuration management systems can automate the configuration task on the application layer. Used in an ERP system supported order-to-cash process, this decouples the service provisioning from expensive technical expertise, and furthers efficiency through automation. Zarnekow and Brenner [17] propose four levels of an IT product (see subsection 2.1). Considering the offerings by Amazon and Salesforce, their pricing models, however, include products only on levels 1 to 3. They have capacity restrictions (e.g. virtual cores or active users), a contract duration, and a price. The same will be the assumption for the SaaS products here.

3 Proposed Solution

The focus of this paper is on the application part of SaaS, but tasks that usually accompany SaaS and are performed by human resources such as initial consulting, service customization, or support are left out.
A reference order-to-cash process for SaaS products could not be identified in the literature. Thus, our process is based on a reference process defined by Schrödl and Bensch [4] that describes how service products are procured. Their work takes a view on services from the consumer's perspective. The described steps are used to derive a simple and generic order-to-cash process for SaaS products.

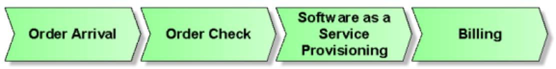

Fig. 1. ARIS value chain diagram: Proposed generic SaaS order-to-cash process

According to Schrödl's and Bensch's model, after a detailed selection process, the procuring party places the order and subsequently monitors its progress, including billing and payment. The provider's equivalent of placing an order is the arrival of an order as depicted by the first process step in **Fig. 1**. In procurement, monitoring is the second step. Schrödl and Bensch assume a detailed communication with the provider prior to order placement. As the procuring party may also be unknown to the provider prior to order placement, the provider has to check the customer's solvency and its own production capacity in the second process step. In line with the procurement process, the third step is the provisioning of the service based on the customer's order which is the focus of this paper. The subsequent billing is the final step.

3.1 From Physical Goods to SaaS Production

We use Scheer's Y-CIM model [27] as a basis to document relevant aspects of the proposed mapping between computer-integrated manufacturing and SaaS production in **Table 1**. The left column contains terms from the Y-CIM model. The right column contains the associated terms from our concept of computer-integrated SaaS production. As the focus lies on the provisioning step, only specific Y-CIM processes from product implementation are included, which are mapped to ITIL's service transition and operation stages. The IaaS tool OpenStack and the configuration management tool Puppet served as cases for our analysis regarding their domains. The service concepts, mentioned in the following, stem from the service models proposed by Übernickel et al. [23] and Ebert et al. [18], which will be discussed in further detail and extended in the next subsection.
A core production concept is the bill of materials, which is mapped to bill of configurations, preliminary service, and IT service. We define a bill of configurations (BOC)

as to contain possibly multiple IaaS instances' configuration models and application configuration models. An IaaS instance's configuration model defines the instance. An instance is a virtual machine that boots with a preconfigured, compressed root disk image. A preliminary service might also be a service performed by a human resource, therefore, only the superior service is called IT service. Work plans are mapped to configuration models. A work plan qualitatively and quantitatively describes how to manufacture semi-finished or finished goods. A configuration model describes how a configuration management tool configures the applications in a running instance. Equipment is mapped to resources. Resources may be information, human, application, and IaaS resources. This is similar to a factory that produces machines for other factories: the same kind of resources that flow into the final IT product are used for producing it (e.g. a configuration management tool is an application that could also be provisioned as a SaaS product).

Table 1. From Y-CIM to computer-integrated SaaS production (CISP)

Y-CIM	CISP: tools and concepts
Bill of materials	Bill of configurations, preliminary service, IT service
Work plan	Configuration model
Equipment	Resources
Product implementation	**ITIL's service transition and service operation**
Control of (C/D)NC-machines and robots	Configuration management tools, IaaS controller
Resource management	ERP system, configuration management database
Warehouse control	IaaS controller, repositories
Materials handling control	Orchestration controller
Maintenance	Software updates (manual/auto-update)
Quality assurance	Monitoring

In the following, the mapping for production implementation processes is described. Control of robots and machines can be mapped to configuration management tools and IaaS controllers. The process of resource management is mapped to an ERP system, which manages human resources, and to a configuration management database (CMDB), which manages application and infrastructure components. The warehouse control process is mapped to the IaaS controller as it controls computing, storage, and networking resources, and is also mapped to repositories that may store application packages and configuration models. The materials handling control is mapped to the orchestration controller, which is introduced in subsection 3.3, maintenance to manual or automatic updates, and quality assurance to monitoring.

3.2 Model of a SaaS Product

In this section, a service model with a strong focus on the application and infrastructure layer is presented. The primary goal is to make the domain knowledge explicit, but the model could also serve as a data model's starting point for use in an ERP system as suggested by Ebert et al. [18]. All models are created with the software ARIS

Business Architect and in accordance with Oestereich's UML 2.5 reference [38]. UML classes are formatted in italics.

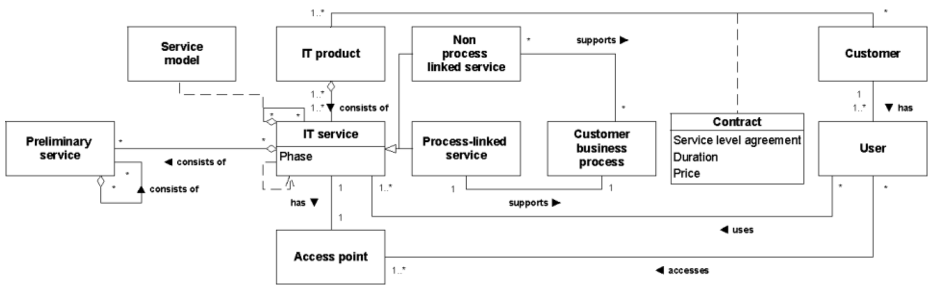

Fig. 2. UML class model: High-level part of SaaS model

The model is based on the high-level view of the models by Übernickel et al. and Ebert et al. [18, 23]. We use a combination of both models, shown in **Fig. 2**, because we do not map individual process activities to resources as Ebert et al. propose. This would be closely related to IT products on level 4 as defined by Zarnekow and Brenner [17], but only IT products from level 1 to 3 are considered here. An *IT product* may consist of multiple instances of an *IT service*, which again may consist of multiple instances of a *Preliminary service* as suggested by Übernickel et al. The UML classes *Service model*, *Contract*, and *User* from Ebert's model still fit. The three included process related classes of Übernickel are reasonably included as well. The class *Contractor* is renamed to *Customer* in order to be consistent. To make it explicit that a *Contract* has a *duration* and a *price* as described in subsection 2.3, the attributes are added. The simplification of not modelling *Service level agreement* (SLA) more comprehensively, as for example done by Brenner et al. [39], is accepted.

In **Fig. 3** the technical level of our SaaS model is described. We modelled the associations that we identified as relevant based on our analysis of Puppet and OpenStack.

A *Preliminary service* uses, possibly multiple, instances of *Resource*. In contrast to the previous high-level service models, *Hardware* and *Network* are replaced with the class *IaaS resource*. Due to the focus of this paper, associations from *Human Resource* and *Information* to other classes were not analyzed. Self-produced preliminary services have bills of configurations, possibly multiple versions, but a preliminary service may also be sourced, not having a bill of configurations.

A *Bill of configurations* contains instance configuration and application configuration models. The model ID could be mapped to version and a full path in a repository. An instance consists of *Computing*, *Storage*, and *Networking* resources. The instance boots with an *Image* of its root disk that contains the *Operating system*, including all necessary drivers and usually a set of applications that are operating system related. It also contains the *Configuration management agent* as well as application configuration options. An *Application configuration option* may be the access rights of a file or the desired state of an operating system level service such as "is running" for a secure shell daemon. The *Instance* itself has multiple possible life-cycle states such as rebooting or terminated. An instance is hosted by a *Node* which is controlled by an *IaaS* *controller*. The *Application state*, which is defined by the *Application configuration model*, has consequences for the general *Instance state*. The *Instance state* also contains an *Instance life-cycle state*.

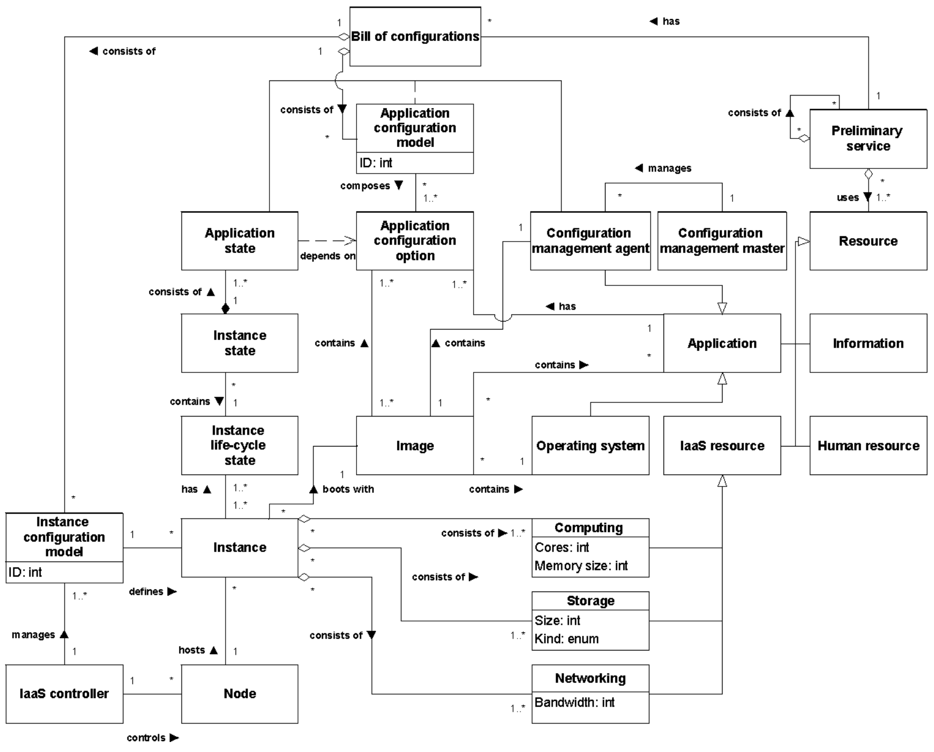

Fig. 3. UML class model: Technical level part of SaaS model
(*Preliminary service* in the top right corner connects the two model parts)

The *Application configuration model* composes multiple instances of *Application configuration option* that the *Application state* depends on. The *Configuration management agent* uses the *Application configuration model* to create a certain *Application state*. The *Configuration master* manages multiple instances via their *Configuration management agent*.

3.3 Components to Support the Provisioning of the SaaS Product

In this subsection the components, in particular their communication, which are necessary for the automation of the SaaS provisioning step, are described. The UML communication model is shown in **Fig. 4**. We use Gronau's work [40] for a generic model of an ERP system. It defines four layers of abstraction: usage, adaptation, application, and data storage. The application layer contains an application server that the business modules finance and accounting, production and logistics, sales, and human resource depend on.

A central component of the presented solution concept is the orchestration controller which can process BOCs that it receives from the production and logistics module of the ERP, get the configuration models from a code repository, and send them to the configuration master and the IaaS controller. The code repository and its controller are used to store application and instance configuration models. The IaaS installation may be operated privately or publicly. This enables scaling to public IaaS providers if necessary.
The communication diagram shows the sequence of calls between the components in the provisioning step of the order-to-cash process; the sequence is indicated by numbers on the edges in the diagram and with parenthesized numbers in the text. Calling components may parameterize their calls and may save return values for subsequent use, as defined in the UML 2.5.

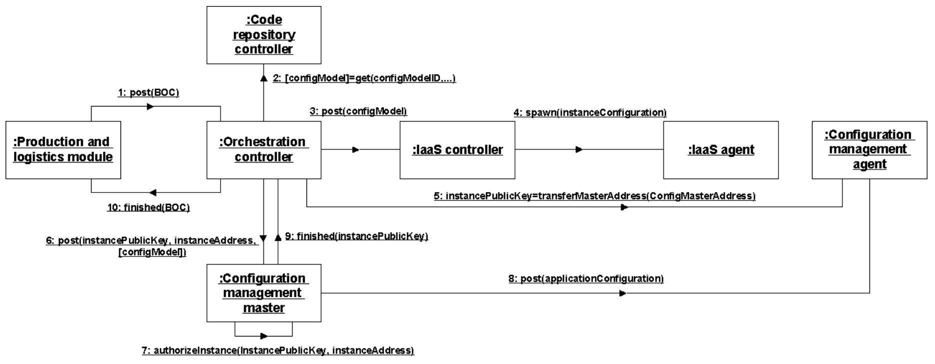
:Code repository controller
2: [configModel]=get(configModelID,...)
1: post(BOC)
3: post(configModel)
4: spawn(instanceConfiguration)
:Production and logistics module
:Orchestration controller
:IaaS controller
:IaaS agent
:Configuration management agent
10: finished(BOC)
5: instancePublicKey=transferMasterAddress(ConfigMasterAddress)
6: post(instancePublicKey, instanceAddress, [configModel])
9: finished(instancePublicKey)
:Configuration management master
8: post(applicationConfiguration)
7: authorizeInstance(InstancePublicKey, instanceAddress)

Fig. 4. UML communication diagram: Components‘ communication during the deployment of one preliminary service with one bill of configurations

After a SaaS product order is passed from the sales module to the production and logistics module of the ERP system, the BOCs are retrieved and sent to the orchestration controller (1). The orchestration controller retrieves a list of configuration models based on their IDs (2) and passes, if applicable, the instance configuration model to the IaaS controller (3), which prompts the node’s IaaS agent to spawn an instance (4). The IaaS controller may set up virtual networks that the instance’s virtual network adapter is connected to. A virtual network may, for instance, be a subnet connected via network address translation to a physical wide or local area network. The orchestration controller then transfers the configuration master’s address to the configuration agent’s instance (e.g. based on predefined IP-address allocation sets) and stores the agent’s public key that authenticates and identifies it (5). The instance agent’s public key, its address, and a list of configuration models are transferred to the configuration management master (6), which then puts public key and address into a whitelist (7) in order to authorize subsequent calls from the agent. The application configurations are subsequently translated to the instance’s operating system and architecture, and are transferred to the configuration agent that configures its instance (8). Using the public key, the master then indicates to the orchestration controller that the configuration task was completed (9). The controller, using its stored mapping of the instance’s agent public key and BOC, communicates to the production and logistics module that the preliminary service associated with this BOC was provisioned (10). The process then continues in the ERP system with the billing process.

4 Evaluation

The model described in the previous section was evaluated by a prototype which verifies that the model is implementable. This corresponds to the approach outlined by Peffers et al. [15]. The prototype was developed in a student project in the context of a lecture. The software products involved are mapped to the process steps in **Fig. 5**.

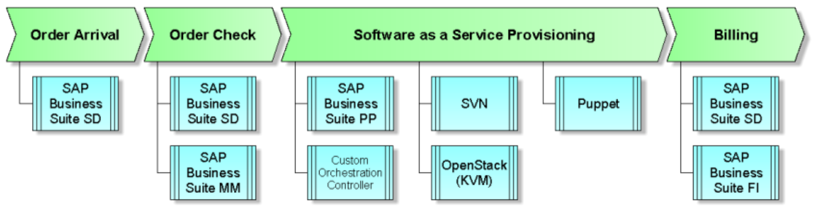

Fig. 5. ARIS value-added chain diagram: Software products in the prototype

The ERP system SAP Business Suite was used and the PP module was programmatically extended so that it can transfer the BOC (which remained a bill of materials in the ERP) with all configuration models in XML to the orchestration controller using a RESTful HTTP web service. The orchestration capabilities are already available in software products, for example in OpenStack's Heat project[1], and furthermore an orchestration specification by OASIS, termed TOSCA[2], exists. We decided to build a simple custom orchestration controller that is able to process the BOC's XML data. The orchestration controller uses the jclouds-API[3], which allows for the management of IaaS resources of a number of systems, including OpenStack and Amazon's EC2. Puppet and OpenStack were used for the prototype. Based on our analysis and on work by Delaet et al. [31], it is reasonable to assume that the prototype could also have been implemented with other configuration management tools such as CFEngine and Chef. The same should be true for OpenStack, considering the capabilities of the jclouds-API.
The customer can compose an IT product based on defined preliminary services and IT services. Prior to offering the service, it must be thoroughly tested in the design and transition stage, whether or not all available combinations work well together. After the order has arrived, it is checked. Solvency can automatically be checked using appropriate SaaS offerings integrated in the SD module. This was not included in the prototype. For the infrastructure part of the service in the MM module, a stock of materials (e.g. virtual CPUs, virtual main memory, virtual storage space) exists,

[1] https://wiki.openstack.org/wiki/Heat
[2] https://www.oasis-open.org/committees/tosca
[3] http://jclouds.apache.org/

which is lessened when an instance is configured and increased as instances are terminated. A stock concept for software only seems reasonable for software that has to be licensed. During the next process step, the SaaS product is provisioned as described above, and in the last step billing commences.
The prototype demonstrates that the transfer of concepts from computer-integrated manufacturing to IT service production is feasible. The model improves both efficiency and effectiveness of IT service provisioning. The automation provides efficiency. With the formalization of configuration knowledge, expensive human resources can be used more effectively elsewhere.

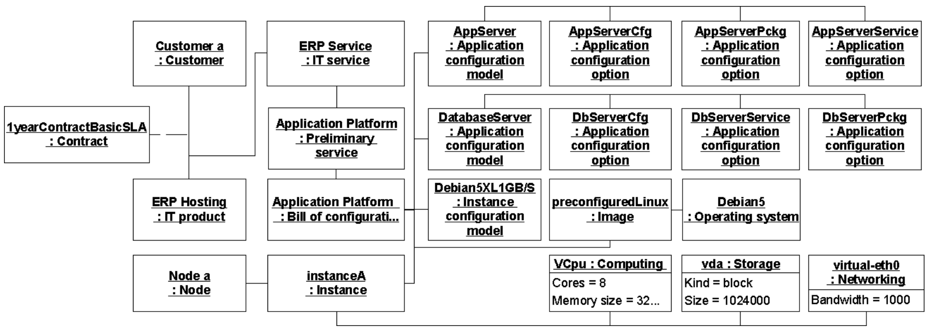
Customer a : Customer
ERP Service : IT service
AppServer : Application configuration model
AppServerCfg : Application configuration option
AppServerPckg : Application configuration option
AppServerService : Application configuration option
1yearContractBasicSLA : Contract
Application Platform : Preliminary service
DatabaseServer : Application configuration model
DbServerCfg : Application configuration option
DbServerService : Application configuration option
DbServerPckg : Application configuration option
ERP Hosting : IT product
Application Platform : Bill of configurati...
Debian5XL1GB/S : Instance configuration model
preconfiguredLinux : Image
Debian5 : Operating system
Node a : Node
instanceA : Instance
VCpu : Computing
Cores = 8
Memory size = 32...
vda : Storage
Kind = block
Size = 1024000
virtual-eth0 : Networking
Bandwidth = 1000

Fig. 6. UML object model: A sample IT product

Fig. 6 shows a sample IT product using a UML object model. The IT product is an ERP SaaS product. The model shows only one preliminary service: the application platform, which is exemplarily broken down in the model. The sizes are specified in megabytes and the bandwidth in megabit per second. The IT service belongs to an IT product which is bought by a customer.

5 Summary, Limitations, and Future Work

In this paper, an order-to-cash process for SaaS provisioning was developed. This contributes to the knowledge base by showing how to automate and integrate IT service provisioning. The process model was implemented in a prototype based on the ERP system SAP Business Suite, and the evaluation verified that the concept is implementable and yields efficiency and effectiveness gains. The model is limited in certain aspects. These limitations may be addressed in future work and are discussed in the remainder of this section.
We focused on the provisioning part of the service and left out stages such as service design. However, in early development states of a service, the architecture can be used as well. Service prototypes can be provisioned and tested very easily. Also, concepts such as automated testing could be incorporated in the design and transition stages of the service life cycle. A custom orchestration controller was developed for the prototype. However, future work could include making the controller compliant with the orchestration specification TOSCA from OASIS, or looking at existing or-

chestration software tools and assess the feasibility of adapting them to the needs of the model. Furthermore, it should be studied how ITIL’s operation stage could be integrated. When using SAP Business Suite for the order-to-cash process it would be obvious to combine it with the SAP Solution Manager, for example, to use its service-desk. In terms of continuous improvement the model should be extended to detail how updated configuration models are to be deployed into production. This would have to be integrated into a rigorous change management process.
Human as well as information resources were ignored in the concept. However, especially for complex SaaS offerings, some tasks which can only be performed by human resources will have to be considered. In the model the infrastructure and applications of SaaS products are configured. However, in an enterprise system context, also process and organizational data in the enterprise system have to be customized. This is usually performed by human resources. Information resources which, for instance, could be customization templates that assist human resources in performing tasks, should be included in the model. Furthermore, human resources are usually required to support users of the SaaS offering. Hallek [21] describes detailed work plans which could be incorporated as well.
SaaS providers were assumed to have private IaaS resources or that these were procured from public IaaS providers. The IaaS concept abstracts from physical hardware. If the IaaS is private, then the ordering of new physical hardware could be linked to the remaining abstracted IaaS material stock in the ERP, and orders could be triggered if stocks reach a certain minimum. The IaaS abstraction should be implementable in an ERP system in the following way: letting the application department define IaaS resources in the ERP system and letting the infrastructure department define physical resources in the ERP system. These physical resources (hardware and network) produce the application department’s IaaS resources. In the case of sourcing IaaS, the physical infrastructure layer would not be represented in the ERP system. Our modelling of the infrastructure resources, particularly the network resource, is too simplistic. This would have to be extended in future work.
Furthermore, we accepted a very simple form of SLA. Options here reach from just letting textual information of the SLA be provided to the responsible people, to letting IaaS controllers automatically decide where to place instances in the IaaS environment according to the SLA’s availability restrictions (e.g. high-availability vs. low-availability IaaS nodes).
The proposed model could also solve the problem of expensively maintaining CMDBs. Information needed in a CMDB could be pulled from the BOC. Every time a service is provisioned the service’s information could be added to the CMDB. This would enable the provider, as depicted in **Fig. 6**, to see which customer has which services in use, where the service is running, and what it consists of.

References

1. Becker, J., Poeppelbuss, J., Venker, D., Schwarze, L.: Industrialisierung von IT-Dienstleistungen: Anwendung industrieller Konzepte und deren Auswirkungen aus Sicht

von IT-Dienstleistern. In: Bernstein, A. Schwalbe, G. (eds.) 10. Internationale Tagung Wirtschaftsinformatik, pp. 345–354. AIS Electronic Library, Zürich (2011)

2. Böhmann, T., Junginger, M., Krcmar, H.: Modular service architectures: a concept and method for engineering IT services. In: 36th Hawaii International Conference on System Sciences (HICSS), pp. 10–20. IEEE Computer Society, Big Island (2003)
3. Marrone, M., Kolbe, L.M.: Impact of IT Service Management Frameworks on the IT Organization - An Empirical Study on Benefits, Challenges, and Processes. Business & Information Systems Engineering. 1, 5–18 (2011)
4. Schrödl, H., Bensch, S.: E-Procurement of Cloud-based Information Systems–a Product-Service System Approach. In: Baskerville, R. Chau, M. (eds.) 34th International Conference on Information Systems (ICIS), pp. 1–19. AIS Electronic Library, Milan (2013)
5. Klaus, H., Rosemann, M., Gable, G.G.: What is ERP? Information Systems Frontiers. 2, 141–162 (2000)
6. Schlichter, B.R., Kraemmergaard, P.: A comprehensive literature review of the ERP research field over a decade. Journal of Enterprise Information Management. 23, 486–520 (2010)
7. Hintsch, J.: ERP for the IT Service Industry: A Structured Literature Review. In: Shim, J., Hwang, Y., Petter, S. (eds.) 19th Americas Conference on Information Systems (AMCIS), pp. 1–9. AIS Electronic Library, Chicago (2013)
8. Zarnekow, R.: Produktionsmanagement von IT-Dienstleistungen: Grundlagen, Aufgaben und Prozesse. Springer, Heidelberg (2007)
9. Goettsch, N., Tosse, T.: Digitale Produktentwicklung und Fertigung: Von CAD und CAM zu PLM. In: Kief, H. B. Roschiwal, H. A. (eds.) CNC-Handbuch 2013/2014, pp. 562-579. Hanser, München (2013)
10. Abramovici, M., Schulte, S.: PLM – Neue Bezeichnung für alte CIM-Ansätze oder Weiterentwicklung von PDM? Konstruktion – Zeitschrift für Produktentwicklung und Ingenieur-Werkstoffe. 1, 64–75 (2005)
11. Gunasekaran, A., Ngai, E.W.T.: Build-to-order supply chain management: a literature review and framework for development. Journal of Operations Management. 23, 423–451 (2005)
12. Stevenson, M., Hendry, L.C., Kingsman, B.G.: A review of production planning and control: the applicability of key concepts to the make-to-order industry. International Journal of Production Research. 43, 869–898 (2005)
13. Gartner, Inc., http://gtnr.it/1qyFdPv (Accessed: 31.07.2014)
14. Hevner, A.R., March, S.T., Park, J., Ram, S.: Design science in information systems research. MIS Quarterly. 28, 75–105 (2004)
15. Peffers, K., Tuunanen, T., Rothenberger, M.A., Chatterjee, S.: A design science research methodology for information systems research. Journal of Management Information Systems. 24, 45–77 (2007)
16. Mell, P., Grance, T.: The NIST Definition of Cloud Computing. Technical report, National Institute of Standards and Technology (2011)
17. Zarnekow, R., Brenner, W.: A product-based information management approach. In: Ciborra, R. Claudio U.and Mercurio, Marco, M. de, Martinez, M., Carignani, A. (eds.) 11th European Conference on Information Systems (ECIS), pp. 2251–2263. AIS Electronic Library, Naples, Italy (2003)
18. Ebert, N., Uebernickel, F., Hochstein, A., Brenner, W.: A Service Model for the Development of Management Systems for IT-enabled Services. In: Hoxmeier, J.A. Hayne, S. (eds.) 13th Americas Conference on Information Systems (AMCIS), pp. 455–462. AIS Electronic Library, Keystone (2007)

19. Bruhn, M., Meffert, H.: Dienstleistungsmanagement als untemehmerische Herausforderung – Eine Einführnng in das Handbuch. In Bruhn, M., Meffert, H. (eds.) Handbuch Dienstleistungsmanagement, pp. 1–25. Gabler, Wiesbaden (1998)
20. Ebert, N.: Produktionsplanung und -steuerung bei IT-Dienstleistern. PhD thesis, Universität St. Gallen (2009)
21. Hallek, S.: Produktionsplanung und -steuerung für IT-Services. PhD thesis, Universität Münster (2009)
22. Pinnow, A.: Das Rechenzentrum als Produktionsstätte für IT-Dienstleistungen - Kapazitätswirtschaft in virtualisierten Rechenzentren. PhD thesis, Universität Magdeburg (2009)
23. Übernickel, F., Bravo-Sánchez, C., Zarnekow, R., Brenner, W.: IS Service-Engineering: A process model for the development of IS services. In: Irani, Z., Sarikas, O.D., Llopis, J., Gonzalez, R., Gasco, J. (eds.), CD-ROM/Online Proceedings of the European and Mediterranean Conference on Information Systems (EMCIS), C29. Costa Blanca (2006)
24. Pilgram, U., Vogedes, A.: Ein Geschäftssystem für ICT-Dienstleister nach industriellen Maßstäben. Zeitschrift HMD – Praxis der Wirtschaftsinformatik. 287, 103—112 (2012)
25. Valiente, M.-C., Garcia-Barriocanal, E., Sicilia, M.-A.: Applying an ontology approach to IT service management for business-IT integration. Knowledge-Based Systems. 28, 76–87 (2012)
26. Ebert, N., Brenner, W.: PPS im IT-Servicemanagement: Möglichkeiten und Grenzen für die Provisionierung standardisierter Services. HMD – Praxis der Wirtschaftsinformatik. 274, 103–111 (2010)
27. Scheer, A.-W.: Wirtschaftsinformatik - Referenzmodelle für industrielle Geschäftsprozesse. Springer-Verlag, Berlin (1997)
28. Spinellis, D.: Don’t Install Software by Hand. IEEE Software. 29, 86–87 (2012)
29. Feitelson, D.G., Frachtenberg, E., Beck, K.L.: Development and Deployment at Facebook. IEEE Internet Computing. 17, 8–17 (2013)
30. USENIX Association, http://bit.ly/1nZC5is (Accessed: 30.07.2014)
31. Delaet, T., Joosen, W., Van Brabant, B.: A survey of system configuration tools. In: Drunen, R. van (ed.) 24th Large Installation System Administration conference (LISA). USENIX, San Jose (2010)
32. Wettinger, J., Behrendt, M., Binz, T., Breitenbücher, U., Breiter, G., Leymann, F., Moser, S., Schwertle, I., Spatzier, T.: Integrating Configuration Management with Model-driven Cloud Management based on TOSCA. In: 3rd International Conference on Cloud Computing and Service Science (CLOSER), pp. 437–446. SciTePress, Aachen (2013)
33. Rackspace US, Inc., http://bit.ly/1pvVnKC (Accessed: 30.07.2014)
34. Forrester Research, Inc., http://bit.ly/1zDPWxC (Accessed: 27.07.2014)
35. León, X., Chaabouni, R., Sánchez-Artigas, M., Garcia-Lopez, P.: Transparently integrating BitTorrent in the data center with smart cloud seeding. IEEE Internet Computing. 18, 47–54 (2014)
36. OpenStack Foundation, http://bit.ly/1u7lLvH (Accessed: 30.07.2014)
37. OpenStack Foundation, http://bit.ly/WRAGiU (Accessed: 30.07.2014)
38. Oestereich, B., Scheithauer, A.: Analyse und Design mit der UML 2.5: Objektorientierte Softwareentwicklung. Oldenbourg Wissenschaftsverlag, München (2012)
39. Brenner, M., Schaaf, T., Scherer, A.: Towards an information model for ITIL and ISO/IEC 20000 processes. In: Raz, D., Schulzrinne, H., Kim, Y.-T. (eds.) 11th IFIP/IEEE Int. Symp. on Integrated Network Management, pp. 113–116. IEEE, Long Island (2009)
40. Gronau, N.: Enterprise Resource Planning – Architektur, Funktionen und Management von ERP-Systemen. Oldenburg, München (2010)