AN ADAPTIVE SOLUTION FOR INTERNET SERVICES' SUPPLY CHAINS

Torsten Hahmann[1]
Jan Möller[1]
Philipp Sommer[1]
Bernhard Peissl[2]
Alexander Wahler[2]

[1] *Hasso-Plattner-Institute for IT Systems Engineering, University of Potsdam, Germany*
{torsten.hahmann|jan.moeller|philipp.sommer}@hpi.uni-potsdam.de
[2] *NIWA Web Solutions, Vienna, Austria*
{bernhard.peissl|wahler}@niwa.at

Abstract. *Industry has adopted service orientation paradigm over the last years. Automatic service discovery, dynamic service composition and process adaptability will bring SOA to its full potential. The integrated research project "Adaptive Services Grid" realizes a platform providing such automation functionalities based on semantics. Enhancing SOA with semantics has a massive impact on the development methodology for the overall system.*

In this paper a business use case for dynamic supply chains demonstrates how to disentangle the interdependencies between service identification, WSDL interface descriptions and their semantic specifications. The paper shows a step-by-step approach for developing such applications on top of adaptive service-oriented architectures. Minimized dependencies between work tasks will allow efficient work distribution among domain experts and service engineers.

1. Introduction

Today's European telecommunication industry is increasingly competitive with many new entrants to the market and a challenging regulatory environment. Along with the ongoing recovery from the technology boom-and-bust, these factors add up to a tough business environment. Price erosion means that providers and operators have realized that they must radically transform the way they do business in order to reduce costs and remain competitive. At the same time, a number of new challenges are emerging, including product innovation, aggressive new market entrants, and the blurring of the boundary between IT and traditional telecommunications. Companies are seeking to grow new business while defending traditional core revenues. The industry suffers from high manpower costs due to a lack of automation, poor time-to-market due to inflexible business processes and poor customer service due to a lack of integrated support systems.

Thus the industry is seeking urgently to reduce IT costs, more than 35% of which are attributable to integration[1]. Furthermore, there is a focus on faster time to market via more flexible business processes and services and a need to reconfigure system components quickly and efficiently in order to satisfy market needs and to provide fully integrated support systems for increasingly sophisticated services.

On the other hand, customers are demanding integrated services, tailored to their specific needs. The market is becoming increasingly federated due both to regulatory pressures and to companies' attempts to catch market opportunities with tailored, bundled services. In this market, the number of B2B relationships between telcos, internet service providers (ISP) and specialist content and service providers has dramatically increased.

All these factors have led many telcos and ISPs to radically rethink the way they operate. They have realized that the new environment requires tighter yet more flexible management of processes and services. In this paper we present the work in developing a B2B service framework for automated reselling of ISPs products through the adoption of "Adaptive Services Grid" (ASG)[2] platform. Focus lies on the used development process that can be applied to application development based on semantically enhanced SOA in general and ASG in particular.

1.1. Terminology

When speaking about services the terminology is essential. As different meanings exist, we use the term *service* here on an abstract business level describing some business capability or in a very general sense of business transactions [3], [5], synonym to the term real-world services used by [1]. When referring to *web services* or *elementary services*, basic functionality in a technical understanding is meant. Definitions differing slightly in scope can be found in [3] and [26]. Nevertheless an elementary service also relates to some business functionality, thus both denotations cannot be clearly separated. A common understanding of services, e-services and web services in their different contexts addresses [4]. We focus on the provisioning of products through a supply chain process. A *supply chain process* in this sense covers the delivery of business services by electronic means. On contrary an *order process*

[1] Gartner Group, 2004
[2] http://asg-platform.org, funded under the 6FP, EC Contract No. 004617

refers only to the collection of customer information relevant for service provisioning.

2. Use Case – Reselling of ISP products

The use case we present here is an example based on the business-to-business (B2B) wholesale model of an Internet Service Provider (ISP). The ISP in our study specializes on products like domain registration and web hosting, not on providing internet access. To understand requirements for a sophisticated B2B solution a look at the existing Business-to-Customer application can be of great avail. By analyzing the current B2C web shop[3] and its underlying provisioning system basic functionality required for the B2B model can be identified.

2.1. Current situation

The present B2C solution based on a web shop allows ordering of domains, emails, and web space. The ISP must provide functionality for domain registration, operating & maintaining DNS information, web hosting configuration, and payment bundled to end-customer products. Selection of domain registration interfaces depends on the specific top-level domain. Assignment of domains with .com or .org endings is governed by ICANN while e.g. national domains are assigned by DENIC in Germany or NICAT in Austria. Registrars accredited by the supervising organizations can register subordinate domains. Web hosting services encapsulate interfaces for web hosting systems. They allow allocation of web space to users while enforcing fine-grained restrictions on data volume, traffic and email configuration.

The goal of developing a B2B solution is the reuse of already available elementary business capabilities. The vision of an enlarged market drives the ISP to shift the existing end-customer-centric application to a more flexible platform that can be used through various front-end solutions operated by resellers. Moreover, service reuse in a flexible environment reduces customer acquisition and support costs for the ISP. The underlying internet service provisioning system requires extension to support a more generalized Business-to-Business approach instead of a restricted B2C application. Currently the complexly interweaved subtasks of product *ordering* in the web shop and *provisioning* of these products through a backend system hinders the reuse of provisioning capabilities through varying order processes.

[3] http://www.chillydomains.com

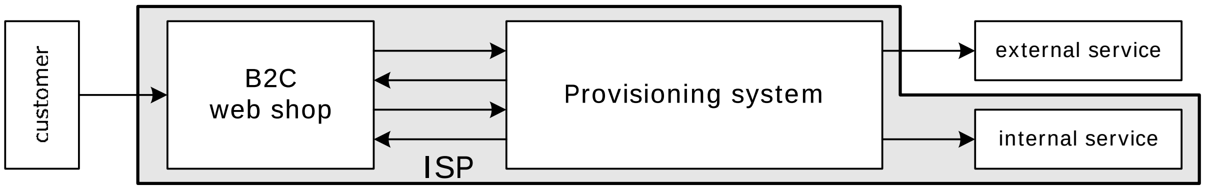


Figure 1: *Current B2C solution*

2.2. B2B solution

Market research shows that there is already a sizable demand for combining domain registration and web hosting services. Especially the association with peregrine products is an interesting market with growth potential. It is anticipated that B2B customers act as resellers that integrate added-value web hosting services (domains, web space, emails) into their existing product portfolio (newspaper subscriptions, broadband internet, community portals). As part of its product bundles, resellers select services offered by the ISP as free add-ons, for bonus programs or as additional features at attractive prices. For example a reseller may order for its customer web space at a special rate or provide them with a domain of their choice when they decide to sign up for a long-term internet access contract. In future only imagination limits evolving reseller models; service provisioning must thus be highly flexible. For the convenience of the resellers, payment services shall be offered as well. Resellers without own billing system or not wanting to deal with payment chooses payment options from the ISP's service pool.

2.3. Business Requirements

The wide range of potential reseller necessitates the development of a solution independent of resellers' order processes and its products. The time-consuming definition of static processes should be avoided. In order to instantly add new resellers that can profit from ISP services provided, reseller integration costs must be reduced to a minimum. If each joining reseller requires high investments in order to deliver customized products, the costs would probably be covered not adequately by expected revenues.

Generally three kinds of flexibility can be identified as necessary for implementing the described business model:

(BR1) an interface allowing fast and cost-efficient integration of resellers with various background and diverse products, resellers want to offer internet services without massive changes to their own application

(BR2) resellers and/or their customers want to customize products requiring a flexible product management and product composition; minimizing the effort required to define/redefine product bundles

(BR3) a flexible extension mechanism to offer new elementary services quickly to all resellers without having to manually change processes

All three types of flexibility together permit reduced time-to-market – essential for business success. Finally business success depends heavily on the ability to ensure maximum availability and in the consideration of quality of services expressed as non-functional requirements that serve the ISP's and reseller's goals best.

Most problematic is the fact that there is no point in time where all possible processes can be designed beforehand. The number of possible product combinations is out of control, static predefined processes cannot cover all combinations within reasonable costs. Dependencies between elementary services will lead to exponentially growing efforts. We derived following technical requirements:

(TR1) separation of order processes from supply chain processes

(TR2) on-demand composition of supply chain processes to cover all possible product combinations and integrate new services automatically

(TR3) adaptive processes for higher availability and consideration of new services at run-time

3. Implementation approaches

Service-oriented computing is a paradigm trying to solve the business requirements BR1 and BR3. By providing high-level interfaces that abstract from concrete operations service-oriented architectures aim to loosen coupling (compare to (TR1)) between components on a service provider's part from those on the service consumer's side [6]. Services in SOA should be reusable and replaceable; it focuses on the scalability for "Internet-scale provisioning and use of services and the requirement to reduce costs in organization to organization cooperation" [14]. The service consumer must compose services in her applications manually. The reference model for SOA does not demand semantics allowing automated service composition [14]. But SOA is only a first step towards rapid application development [10], other research projects like the

Web Service Execution Environment (WSMX) tackle the questions of dynamic selection and semantic web services invocation using ontology mediation [8], [15]. The ASG project goes beyond dynamic service selection [18] by offering advanced service composition capabilities [10] – merging advantages of the workflow and the AI approach to allow on-demand workflow generation based upon AI planning algorithms [16], [17], [20].

3.1. Adaptive Services Grid

"The goal of Adaptive Services Grid (ASG) is to develop a proof-of-concept prototype of an open platform for adaptive services discovery, creation, composition, and enactment." [2]. It extends the concept of service oriented architectures by formal semantics for service specifications as WSMX does. The ASG reference architecture explains in detail how ASG achieves automatic composition of semantic web services [10]. Basically a user sends a request containing initial state and desired goals to the platform. The life-cycle of the service provisioning triggered by the user request spans three sub-cycles.

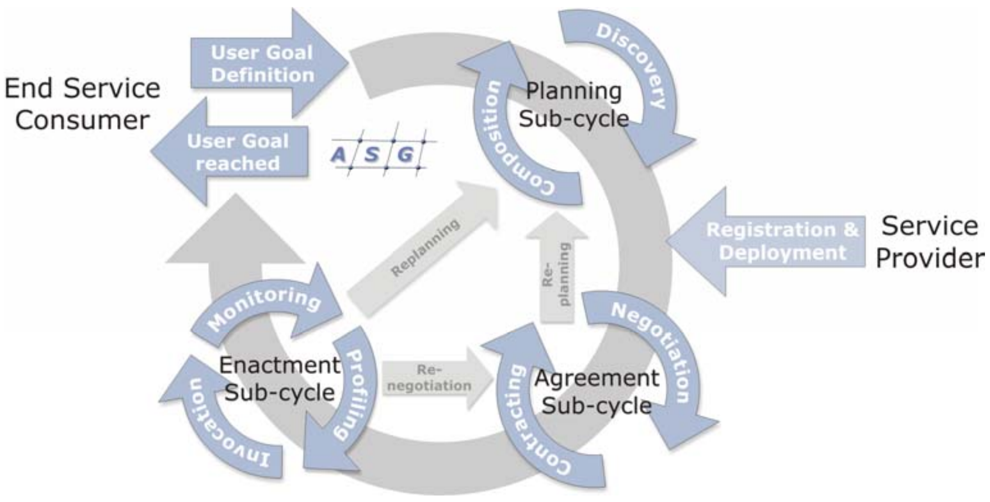


Figure 2: *ASG service provisioning life-cycle [10]*

In the *planning sub-cycle* ASG uses reasoning to discover appropriate services from the service landscape to fulfill user requests. We must distinguish between two ASG-specific types of services. *Atomic services* encapsulate a coherent piece of semantically described functionality whereas *composed services* are created at runtime and combine functionality of arbitrary atomic services. Due to the lack of machine-processable semantics in earlier architectures composition occurs manually or semi-automatic at design time [22], [23]. Formal semantic specifications enable the ASG platform to be highly adaptable to changes in the service landscape. New services can instantly been selected when composing processes. In the context of Internet services supply chains the reseller expresses its goals and ASG composes provisioning processes on-demand – resolving (TR2) through a very flexible mechanism.

Semantic service capabilities are bound to concrete atomic services in the *agreement sub-cycle*. Atomic services are selected by evaluating quality of service requirements expressed by the reseller or ISP, e.g. execution costs. Negotiation allows contracting with candidate services resulting in service level agreements (SLA). The outputs are BPEL-compliant process definitions executable with any workflow engine supporting BPEL standards [25]. *Enactment sub-cycle* uses a BPEL-engine to invoke negotiated service implementations. Endpoint references identify atomic service instances deployed on the ASG grid service infrastructure [27] according to WS-Addressing standard [6]. Monitoring allows detection of failures and violations of service level agreements contracted with atomic services. In such cases, re-negotiation of elementary services with equivalent capability is triggered. If no alternative services fulfill demanded requirements, re-planning takes place in order to adapt the initially planned process to service availability [10]. E.g. in case of failure of a selected payment service provider another provider can be chosen in our use case.

Dynamic service composition and adaptability (see (TR2), (TR3)) satisfy the business requirements (BR1) and (BR2) requesting flexible ways to create supply chain processes for varies resellers and product combinations. Overall availability of business services offered also increases significantly. Earlier approaches to automatic service composition left execution possibilities for composed service open [17], [19], [20]. However, benefits of automatic service composition for application integration are of limited practical use without facilities covering service execution [2], [30]. Compared to traditional architectures, SOA and WSMX, the ASG platform is clearly ahead in the fields of automated and dynamic service composition, adaptability and overall flexibility. These aspects are most important when choosing a platform suiting our business requirements extracted in part 2.3.

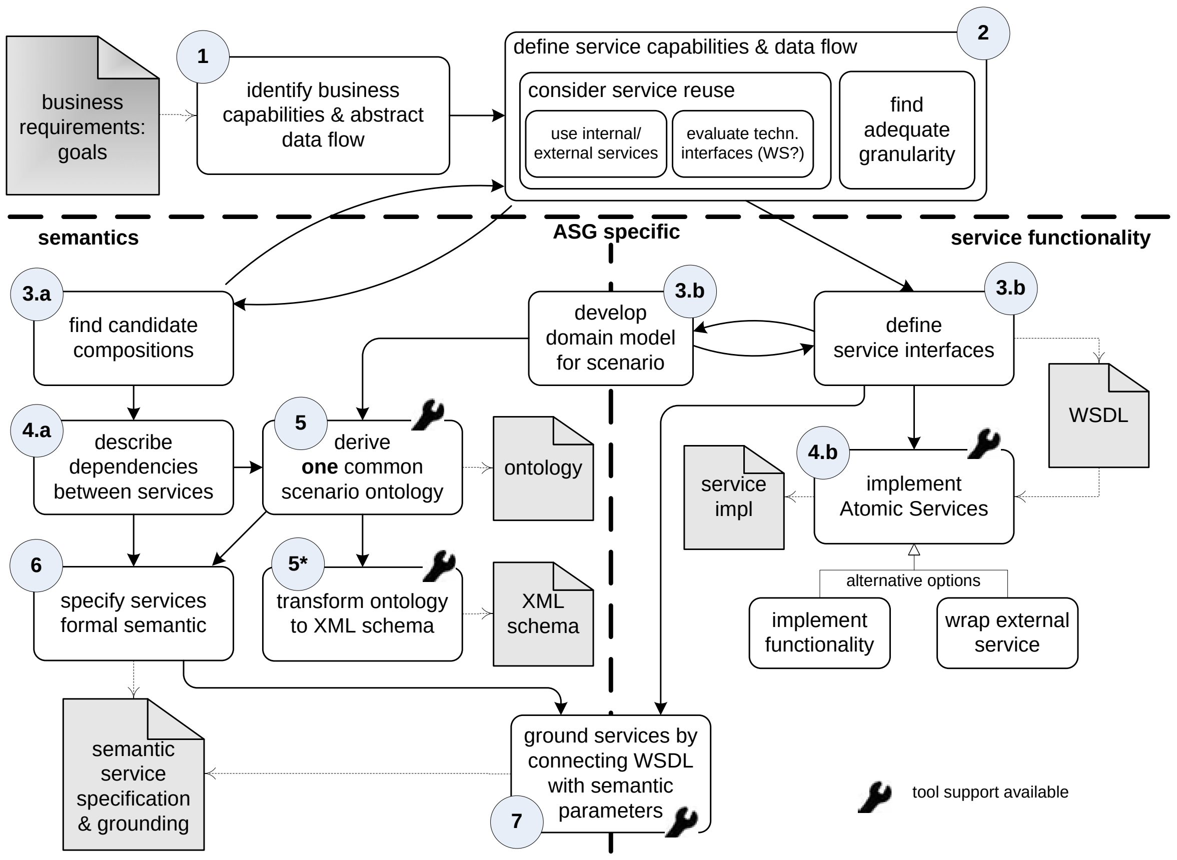
4. Development of an ASG-based solution

A roadmap for application development in the context of ASG must regard interdependent aspects like domain ontology, elementary service identification, semantic service specifications, service groundings and service implementation.

The development process starts with clearly defined business requirements of a proposed application. Following artifacts are expected as results:

1. Interface definitions (WSDL) of elementary services and their implementation as atomic services for the grid infrastructure
2. Domain ontology describing concepts of the use case that can be referred to in the semantic service specifications and their representation as XML schema for use in WSDL
3. Semantic service specifications for service discovery and their service groundings for mapping semantic parameters to WSDL parameters

The description of the service landscape is not a goal of the methodology itself, but can be of great help through the whole development process. It evolves over time, beginning with rudimentary capability descriptions. Successively domain model, concrete in- and output data and informal pre- and post-conditions supplement the service landscape. A consistent service landscape enhances knowledge transfer between participants – as it collects knowledge from both domain experts and service engineers. In particular, it proved helpful for the steps ontology definition and semantic service definition.



business requirements: goals
1
identify business capabilities & abstract data flow
2
define service capabilities & data flow
consider service reuse
use internal/ external services
evaluate techn. interfaces (WS?)
find adequate granularity
semantics
ASG specific
service functionality
3.a
find candidate compositions
3.b
develop domain model for scenario
3.b
define service interfaces
WSDL
4.a
describe dependencies between services
5
derive one common scenario ontology
ontology
service impl
4.b
implement Atomic Services
alternative options
implement functionality
wrap external service
6
specify services formal semantic
5*
transform ontology to XML schema
XML schema
semantic service specification & grounding
ground services by connecting WSDL with semantic parameters
7
tool support available

Figure 3: *Methodology applied in the dynamic supply chain use case*

When developing an ASG-based solution two parts of the methodology must be strictly separated. The first part containing step 1 and 2 in **Figure 3** is common to all service oriented architectures: main task is to define service capabilities in an appropriate granularity. Service reuse advocated by SOA presses us to take preexistent services into consideration. How primary business services and shared services are discovered and high-level interactions identified is approached in [11]. SOA methodology heavily depends on the business circumstances. We concentrate on the steps specific to an ASG-based solution, starting with the defined service capabilities as result of step 2 (see **Figure 3**). We separate it into two independent tasks: (I) defining and implementing service functionality (3.b, 4.b) and (II) providing formal semantics for dynamic service composition (3.a to 6). Finally we must ground semantic specifications to service implementations, see step 7 in **Figure 3**.

4.1. Service identification

Here the term service identification covers the detection of well-shaped functionalities elementary services provide. Most important is the balance between highly reusable functionality and strongly de-coupled software components containing inseparable logic. Granularity and composability of individual services must be evaluated according to specific business requirements. Distinct service types are characterized by its interfaces (web service vs. legacy interface) and its service provider (internal vs. external). In the case of internet services many services like web hosting or national payment options are provided by the ISP itself. All services for domain registration and some of the standardized payment methods are offered using external providers. Nevertheless, even external services offering standardized web service interfaces cannot be simply plugged into ASG. Services use specific properties and reference IDs in ASG and require specific deployment descriptors – the term *atomic service* refers exactly to such an adopted service. It can encapsulate external web services, use legacy interfaces or implement business logic itself.

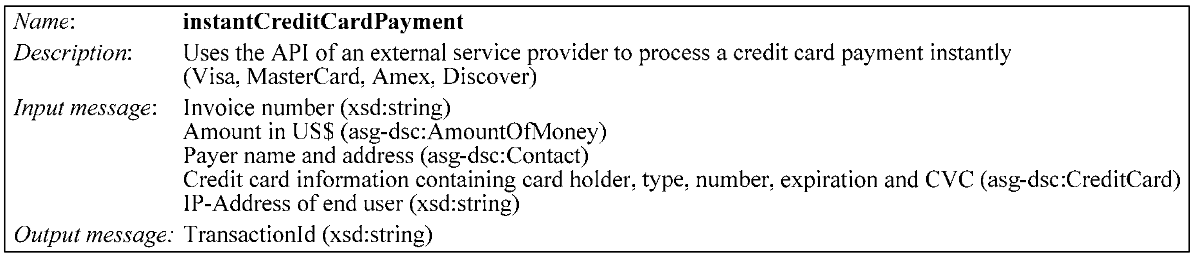
Name:	**instantCreditCardPayment**
Description:	Uses the API of an external service provider to process a credit card payment instantly (Visa, MasterCard, Amex, Discover)
Input:	amount due, credit card information
Output:	success

Figure 4: *abstract elementary service description for service landscape*

Service identification results in the description of services in a textual representation, defining functionality, input-/output messages (parameters can be mutually exclusive) by high-level means as seen in **Figure 4**. These short descriptions must be supplemented by detailed documentation of external or otherwise preexisting functionality. Foreseeable exceptions leading to final states or meaningful intermediary states (processable in renegotiation or replanning) must be captured as well.

4.2. Service functionality

The interface descriptions containing input and output of services must be refined with precise data types. Outputs of this task are service operations with respective messages as WSDL definitions [7]. From our use case development we recommend using an iterative approach to model data types with UML class diagrams and concrete WSDL messages. Service engineers give feedback when describing WSDL messages to the domain engineering experts, who likewise return feedback to the service engineers. Data types should cover different services with similar functionality, e.g. services of two payment providers should use common data types like “CreditCard” or “AmountOfMoney” encouraging reuse. With regard to ontology definition such general concepts have to be identified. In [8], [23] and [28] methodical concepts for ontology-based knowledge management are discussed in-depth. The resulting service landscape expresses input and output parameters as complex data types defined by the corresponding domain model. Some parameters not mentioned in the service landscape from step 2 are added because of requirements of external service providers, e.g. one of our payment services asks for the end customer’s IP-address in order to prevent credit card fraud.



Name:	**instantCreditCardPayment**
Description:	Uses the API of an external service provider to process a credit card payment instantly (Visa, MasterCard, Amex, Discover)
Input message:	Invoice number (xsd:string) Amount in US$ (asg-dsc:AmountOfMoney) Payer name and address (asg-dsc:Contact) Credit card information containing card holder, type, number, expiration and CVC (asg-dsc:CreditCard) IP-Address of end user (xsd:string)
Output message:	TransactionId (xsd:string)

Figure 5: *concrete in- and output messages added to elementary service description*

4.3. Semantics

The methodology for defining ASG-specific semantics goes beyond tasks suiting service-oriented computing and web service development in general. It concentrates on the artifacts *ontology* and *semantic service specifications*. For tackling the semantics part, the rudimentary service landscape as described in 4.1 and the domain model developed in step 3.b are required as inputs.

For transforming the domain model to a corresponding ontology, dependencies between services must be taken into account. Such *conditions (*consisting of *preconditions* and positive as well as negative *effects)* cannot be expressed in the definitions of messages and endpoints written in WSDL, thus conditions must be part of the formal semantics. By trying to compose services manually (step 3.a in **Figure 3**), domain experts can identify conditions that limit services to a specific execution order (see example composition in **Figure 6**). Our experience indicated that even a small selection of service compositions helps tremendously in uncovering a high percentage of all semantic dependencies between services. For much larger service pools, this must be investigated further. The process of "playing service composer" also helps detecting similarities in conditions between interchangeable services. Only the right level of abstraction for specifications will allow on-demand negotiation with service instances based on a specific set of given preconditions and desired effects.

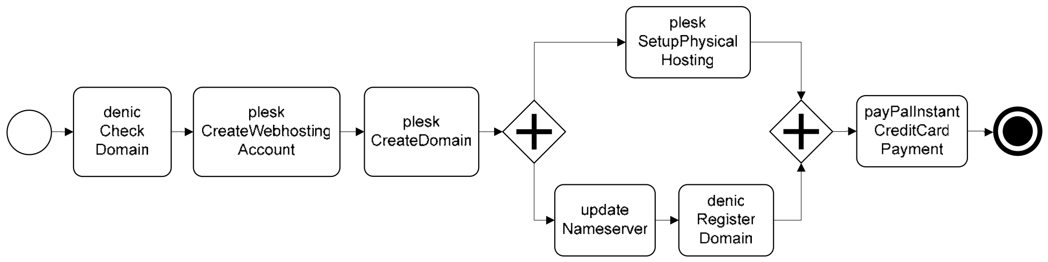

denic Check Domain
plesk CreateWebhosting Account
plesk CreateDomain
plesk SetupPhysical Hosting
update Nameserver
denic Register Domain
payPalInstant CreditCard Payment

Figure 6: *exemplary service composition*

In the ontology data types from the domain model usually map to *concepts*, associations between concepts are expressed as *relations*. Complex data types must be decomposed to reusable concepts and relations between concepts. In the long term ASG aims to use WSMO as framework for specifying ontologies and web services semantically [21]. Since groundings are not yet definite in WSMO [12], we did not use WSML for our semantic service specifications. Thus, we formulate our semantic service specifications and ontology in the object-oriented knowledge-base language Flora2 [29] – comparable to previous ASG prototype implementation efforts [2], [13].

For our use case an open platform (also referred to as *marketplace*) allowing everybody to publish services using their own ontologies is a goal too zealous and not fitting investigated business needs. We assume one common ontology whose concepts are shared by all services. This pragmatic approach is more likely to guarantee seamless interaction between services as the ISP controls all artifacts: semantic service specification, service interfaces and ontology. Devising an integrated ontology will ensure consistent data flow during process enactment. However, an integrated ontology usually consists of several partial ontologies describing aspects of the application domain.

Name:	**instantCreditCardPayment**
Description,Input, Output: *unchanged*	
Precondition 1:	valid input parameters
Effect 1:	the given creditCard has been charged the stated amount of money

Figure 7: *preconditions and effects added to elementary service description*

In ASG each atomic service is assigned exactly one semantic specification and one service grounding. A semantic service specification can contain several sets of conditions that are used for reasoning on semantic services during planning sub-cycle (see **Figure 2**). Negotiation with potential atomic service instances rests upon a single set of conditions, not upon the whole semantic service specification. More than one semantic service specification can contain congruent sets of preconditions and effects.

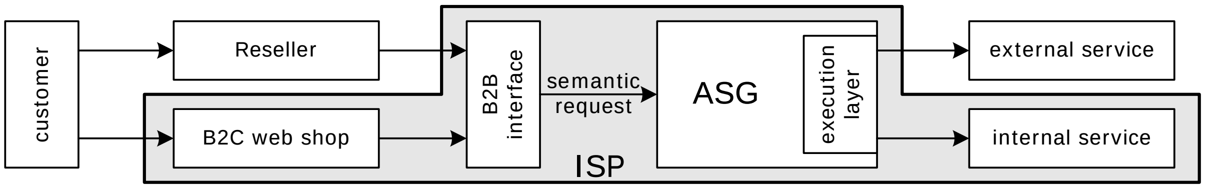


Figure 8: *B2B solution using ASG as adaptive service platform*

5. Conclusion

Our business requirements demanded high flexibility for combining elementary services to various end customer products. We have reached that goal by using the dynamic service composition, negotiation and enactment features of ASG. All ASG-based solutions require the artifacts discussed in our methodology – we have given them a suitable and efficient order and minimized feedback loops and identified precise document-oriented work steps.

The methodology presented here refines the method discussed in [13] by a gradually approach to defining the service landscape. It has been applied in our use case successfully. The development of the use case showed that the perceived complex dependencies between different artifacts must be disentangled in order to provide ASG-users with a methodology of practical use. Our artifact-driven method defines specific steps to develop all aspects of our scenario – both, WSDL-interface descriptions and their semantic specifications. This pragmatic approach comparable to a cookbook appeared much more usable when all parts of an ASG use case are controlled by a single stakeholder. It is important to understand that service landscaping is a process covering all phases of the development. The service landscape is neither a fixed artifact that can be defined in a single subtask nor a primary output.

We have identified three tasks for developing an ASG-based solution: (I) service identification and (II) service functionality definition/implementation – both generally required for service-oriented computing and (III) specification of semantics in order to allow dynamic selection and composition of services to complex processes. All platforms using semantic web services must devise methods for defining semantic specifications, ontologies, and service groudings.

The method we applied leaves out mediation – an acceptable limitation when targeting a closed platform addressing rapid application development.

6. Outlook

We have already followed most of the steps for a selected set of services – registration services and payment services. For these services a complete service landscape has been developed and we have produced all necessary artifacts: WSDL-interfaces, atomic service implementation (wrapping external services), semantic service specifications and partial ontologies. The ongoing work focuses on adding more services, especially web hosting services and on setting up a prototype system for demonstration purposes.

In future, two types of services must be distinguished: product-supply-services and services for cross-cutting-concerns as payment, authorization and notification. These must be generalized to avoid exponential extension of conditions applying to product-supply-services. E.g. all services involving monetary costs must ensure proper payment. However, trusted resellers using in-house billing can order these products without direct payment. How such complex scenarios can be modeled with ontology and conditions is left open. A solution could be the use of non-functional properties e.g. for describing payment requirements. In further research the questions of ontology mediation as well as testing correctness and completeness of semantic services must be tackled.

Here we looked at the development of one special use case scenario. However, in the future it must be analyzed how the proposed methodology can be applied to use cases with similar requirements. Especially larger applications might demand a formalized document-flow to support distributed work of several specialists.

7. Acknowledgement

We like to thank all ASG partners, in particular niwa Web Solutions, Vienna and all participants of the common Vienna ASG C6/C7 work component meeting in November 2005 for their valuable input. We especially appreciate the guidance and feedback of our supervisors Guido Laures, Harald Meyer and Peter Tröger at Hasso-Plattner-Institute Potsdam (HPI) and we want to thank all contributors to the ASG student-project in the HPI bachelor program for the good cooperation.

8. References

[1] Akkermans, H., Baida, Z., Gordijn, J., Peña, N., Altuna, A., Laresgoiti, I. 2004: Value webs: Using ontologies to bundle real-world services. IEEE Intelligent Systems - Semantic Web Services, 19(4) 57-66

[2] ASG consortium 2005: Adaptive Services Grid, www.asg-platform.org

[3] Austin, D., Barbir, A., Ferris, C., Garg, S. 2004: Web services architecture requirements, W3C working group note, http://www.w3.org/TR/wsa-reqs

[4] Baida, Z., Gordijn, J., Omelayenko, B., Akkermans, H. 2004: A Shared Service Terminology for Online Service Provisioning, Proceedings of ICEC 2004. 1-10

[5] Berry, L. 1981. Perspectives on the retailing of services. In Stampfl, R. W. and Hirschman, E. C., editors, Theory in Retailing: Traditional and Non-traditional Sources.

[6] Box, D., Curbera, F. (editors) 2004: Web Services Addressing (WS-Addressing), http://www.w3.org/Submission/ws-addressing/

[7] Chinnici, R., Gudgin, M., Moreau, J.-J., Schlimmer, J., Weerawarana, S. 2003: Web Services Description Language (WSDL), Version 2.0, http://www.w3.org/TR/2003/WD-wsdl20-20031110/

[8] Grüninger, M., Fox, M.S. 1995: Methodology for the desing and evaluation of ontologies. Workshop on Basic Ontological Issues in Knowledge Sharing at IJCAI 95.

[9] Haller, A., Cimpian, E., Mocan, A., Oren, E., Bussler, C. 2005: WSMX - A Semantic Service-Oriented Architecture, Proceedings ICWS 2005. 321-328

[10] Jank, K., Laures, G. 2005: Reference Architecture: Requirements, Current Efforts, and Design. Adaptive Service Grid Deliverable D6.V-1.

[11] Jones, S., Morris, M. 2005: A Methodology for Service Architectures. OASIS draft 26.Oct.2005.

[12] Kopecký, J., Roman, D. 2005: D24.2v0.1. WSMO Grounding, WSMO Working draft. http://www.wsmo.org/2005/d24/d24.2/v0.1/20050119/#grounding_wsdl

[13] Laures, G., Meyer, H., Breest, M. 2005: An Engineering Method for Semantic Service Applications. First International Workshop on Design of Service-Oriented Applications (WDSOA 05) at ICSOC 05.

[14] MacKenzie, C., Laskey, K., McCabe, F., Brown, P., Metz, R., OASIS 2005: Reference model for service oriented architectures. Working draft 09.Sept.2005.

[15] McIlraith, S., Son,T.C., Zeng, H. 2001: Semantic Web Services. IEEE Intelligent Systems, Special Issue on the Semantic Web. 16(2) 46-53

[16] Meyer, H. 2005: Entwicklung und Realisierung einer Planungskomponente für die Komposition von Diensten, Diploma thesis at HPI.

[17] Milanovic, N., Malek, M. 2004: Current Solutions for Web Service Composition, IEEE Internet Computing. 8(6) 51-59

[18] Paolucci, M., Kawamura, T., Payne, T., Sycara, K. 2002: Semantic Matching of Web Service Capabilities. ISWC 2002, 333–347

[19] Pistore, M., Bertoli, P., Barbon, F., Shaparau, D., Traverso, P. 2004: Planning and Monitoring Web Service Composition. Proceedings of AIMSA 2004. 70-77

[20] Rao, J., Su, X. 2005: A Survey of Automated Web Service Composition Methods. SWSWPC 2004. 43-54

[21] Roman, D., Keller, U., Lausen, H., de Bruijn, J., Lara, R., Stollberg, M., Polleres, A., Feier, C., Bussler, C., Fensel, D. 2005: Web Service Modeling Ontology. Applied Ontology, 1(1) 77-106

[22] Sirin, E., Hendler, J., Parsia, B. 2002: Semi-automatic Composition of Web Services using Semantic Descriptions, Web Services: Modeling, Architecture and Infrastructure workshop in conjunction with ICEI 2003.

[23] Staab, S., Studer, R., Schnurr, H.P., Sure, Y. 2001: Knowledge processes and ontologies. IEEE Intelligent Systems. 16(1) 26-34

[24] Talib, M.A., Yang, Z., Ilyas, Q.M. 2005: A framework towards Web services composition modeling and execution. Proceedings of IEEE EEE05 international workshop on Business services networks. 51-59

[25] Thatte, S. (editor) 2003: Business Process Execution Language for Web Services, Version 1.1.

[26] Tidwell, D. 2000: Web services – the web's next revolution, IBM tutorial. www.ibm.com

[27] Tröger, P., Böhme, H., Polze, A.: ASG Services Grid Infrastructure. Adaptive Service Grid Deliverable D5.III-1.

[28] Uschold, M. 1996: Building Ontologies: Towards a Unified Methodology. Proceedings of BCS SGES 96.

[29] Yang, G., Kifer, M., Zhao, C., Chowdhary, V. 2005: Flora-2 User's Manual, http://flora.sourceforge.net/docs/floraManual.pdf

[30] Zeng, L., Benatallah, B., Lei, H., Ngu, A., Flaxer, D., Chang, H. 2003: Flexible Composition of Enterprise Web Services, Electronic Markets - The International Journal of Electronic Commerce and Business Media. 13(2), 141-152