


An architectural perspective on service adoption: A platform design and the case of pluggable cross-border trade compliance in e-commerce

Fabian Aulkemeier, Maria-Eugenia Iacob , and Jos van Hillegersberg

Centre for Telematics and Information Technology, University of Twente, Enschede, The Netherlands

ABSTRACT

Current e-commerce architectures rely on a small number of monolithic application systems. The adoption of innovative IT functionality within these architectures is a tedious and complex task. The purpose of this article is to present a design for a novel platform architecture to improve the pluggability of e-commerce services. More precisely, the pluggability of services in current architectures is described, a platform architecture is introduced, and its effect on pluggability is evaluated by means of a prototype. The enablers for this architecture are the recent innovations of web programming interfaces, delegated resource access, and client-side web application frameworks. Those technologies are common in social media and their potential in enterprise computing is revealed in this work, based on the example of a trade compliance service for cross-border retail. The results of this study suggest that the architectural design has merits as it improves the pluggability of e-commerce services. Hence, e-commerce companies should consider a paradigm shift and move from using self-contained application components to a platform-based adoption of complementary services.

KEYWORDS

Cloud platform; e-commerce services; enterprise architecture design; microservices; pluggability

1. Introduction

E-commerce is becoming an increasingly important channel for retailers of any size and sector. By using specialized information technology (IT) solutions, retailers can improve service quality and information quality, two of the main drivers for trust and customer loyalty (Yoon and Kim 2009). Furthermore, the domain is characterized by the criticality of interorganizational collaboration with suppliers, partners, and customers, which must be projected onto the IT architecture. Cloud-based services are a promising approach to adopt specific functionality and to build an IT landscape with increased interoperability capabilities (Dhar 2012).

Despite the potential benefits of the service-based approach, the adoption of new services can be cumbersome. As Kephart and Chess (2003) state, the complexity of a system is higher than the complexity of the entirety of services. Thus, system complexity depends on the ability of the services to manage their own operation within the system and to reduce the requirements for system integrators to handle the complexity, originating from a growing amount of services. Today's cloud-based services are a step forward to an easy adoption of individual pieces of functionality. However, as Martens, Walterbusch, and Teuteberg (2012) point out, the required resources for adopting cloud services are often underestimated as the need for evaluation, implementation, configuration, and integration gets neglected.

CONTACT Fabian Aulkemeier  f.m.aulkemeier@utwente.nl  Centre for Telematics and Information Technology, University of Twente, CTIT, P.O. Box 217, 7500 AE Enschede, The Netherlands.

Color versions of one or more of the figures in the article can be found online at www.tandfonline.com/hocce.

© 2017 2017 Fabian Aulkemeier, Maria-Eugenia Iacob, and Jos van Hillegersberg. Published with license by Taylor & Francis. This is an Open Access article distributed under the terms of the Creative Commons Attribution-NonCommercial-NoDerivatives License (<http://creativecommons.org/licenses/by-nc-nd/4.0/>), which permits non-commercial re-use, distribution, and reproduction in any medium, provided the original work is properly cited, and is not altered, transformed, or built upon in any way.

The concept of pluggability focuses on the capabilities of a service to facilitate its own adoption within a system (Aulkemeier, Iacob, and van Hillegersberg 2015). It is defined as a quality characteristic of software services and is to be assessed from a user perspective. It reflects the criteria affecting the need for technical skills and labor to consume a service. More precisely, investigating pluggability means breaking down the use of a service into various phases, such as deployment or integration, and to consider the required time and expertise during each phase. To cope with the increased number of IT systems, service providers should aim for a high pluggability for their products. Otherwise, the ownership of a multitude of services would generate too much overhead, thus leading to an inflexible infrastructure.

Platforms have been promoted as a means to increase the quick connect capability within a network (Heck and Vervest 2007). The provisioning of services through a platform means bundling the reoccurring resources and features across application components, and making them accessible for reuse. Following this principle, the platform contains the stable components and supports the variability and evolvability of the other services by constraining their linkages (Baldwin and Woodard 2009). Eventually, it can help retailers decrease efforts related to service adoption, and allows service providers to offer services that are easier to consume.

The research question we answer in this work is, how can a platform architecture for e-commerce increase the pluggability of services. Existing research on platform architectures focuses merely on facilitating specific phases of service development or service consumption. Platforms for service provisioning include Software-as-a-Service (SaaS) marketplaces (Giovanoli, Pulikal, and Grivas 2014) or service directories (Maximilien, Ranabahu, and Gomadam 2008). Another category of platforms is dedicated to the integration of services. These act as intermediaries between services, and manage the access to web services or application programming interfaces (APIs) of the underlying software components (Weerawarana et al. 2005). Recent work on Platform-as-a-Service is mostly oriented toward software development and operation and favors the needs of service implementers (Paraiso et al. 2012). Other platforms exist that allow implementation services based on open data models, but are not applicable for private data in an e-commerce context (Pedrinaci et al. 2010). In contrast to the existing work within the broad body of research in software platforms, we focus on the use of platforms in a business information system context. More specifically, we present a software service platform that facilitates the adoption of e-commerce services for online retail.

In this research, we follow the design science research methodology (DSRM) by Peffers et al. (2007), which provides a framework for producing and presenting design artifacts in the field of information systems. It prescribes five phases, namely 1) motivation, 2) objective definition, 3) design, 4) demonstration, and 5) evaluation. The design artifact and the main contribution of this paper are a platform architecture to facilitate pluggable e-commerce services. The architecture is represented by means of an ArchiMate model (The Open Group 2016). Its objective is to solve the problem of low pluggability of current e-commerce services. It draws upon recent technologies and protocols originating from the social media and mobile service industry, and applies them in an enterprise information system context.

To evaluate the design artifact, we consider the business case of an e-commerce company that plans to get involved in cross-border selling to reach a larger customer base. A key hindrance for cross-border e-commerce lies in the diversity of national legal regulations (Lendle et al. 2012). Based on a prototypical implementation of the platform, we present a solution for a pluggable service that ensures trade compliance in cross-border retail. More precisely, the service helps retailers correctly calculate the foreign tax and customs during a sales transaction. The evaluation of the architecture is carried out through the comparison, in terms of pluggability, of an existing reference solution with the platform-based service.

The paper is structured according to the DSRM. In the next section, we present the business case by means of an existing reference architecture for a cross-border trade compliance service, originating from a service provider in the field (Section 2.1). We then assess the reference architecture with regard to the six phases of service adoption. The findings are discussed in Section 2.2 and provide a

motivation for a novel platform-based solution. In Section 2.3, we elaborate on the concept of pluggability as the main objective for the design. In Section 3, we present the platform architecture and describe how it improves pluggability. The prototypical implementation of the architecture and the pluggable trade compliance service are presented and evaluated in Section 4.

2. Adoption of e-commerce services

The purpose of this section is to specify the problems related to service adoption and to motivate the need for an architecture that facilitates the provisioning of services for e-commerce in a pluggable manner.

2.1. The trade compliance case

For a better understanding of the problems related to service adoption, and the capabilities of our design, we consider a trade compliance service that helps online retailers expand their e-commerce business cross-border. The functional capabilities of common e-commerce solutions do not allow online shops to manage legal regulations for cross-border selling in a straightforward manner (Aulkemeier et al. 2016b). By addressing this challenge through a pluggable service, we can demonstrate the advantages of the architecture regarding the adoption of services, and its benefits in embracing novel business opportunities.

The objective of the service is to provide real-time updates of orders with costs for international shipment, customs, and value added taxes (VAT). In this way, the buyer gets accurate information on the total costs of his/her online purchases from foreign retailers. To date, those costs are often claimed by tax authorities during the transit. Figure 1 illustrates the anticipated solution.

The customer can receive information about all of the costs related to the cross-border transaction. Furthermore, from a retailer perspective, the correct application of tax regulations often depends not only on a single transaction, but also on the total value of goods sold to a specific

The screenshot shows a checkout page for 'Electronic Mall'. The page title is 'Verify your order'. It displays two items in the cart: a '15 inch laptop by HP' and a 'Laptop charger by HP'. Below the cart, there is a table for shipping and tax information, and a final total amount of 1856.62 €.

Product	Quantity	Price	Total
15 inch laptop by HP Status: In Stock	1	1,439.90 €	1,439.90 €
Laptop charger by HP Status: In Stock	1	59.50 €	59.50 €

Name	John Smith	Subtotal	1499.40 €
Street	Random Street 15	Shipping	6.05 €
Postcode	12345	VAT	314.88 €
City	San Francisco	Customs	36.29 €
Country	United States	Total	1856.62 €

Buttons: Continue Shopping, Place order

Figure 1. Checkout with tax and customs information.

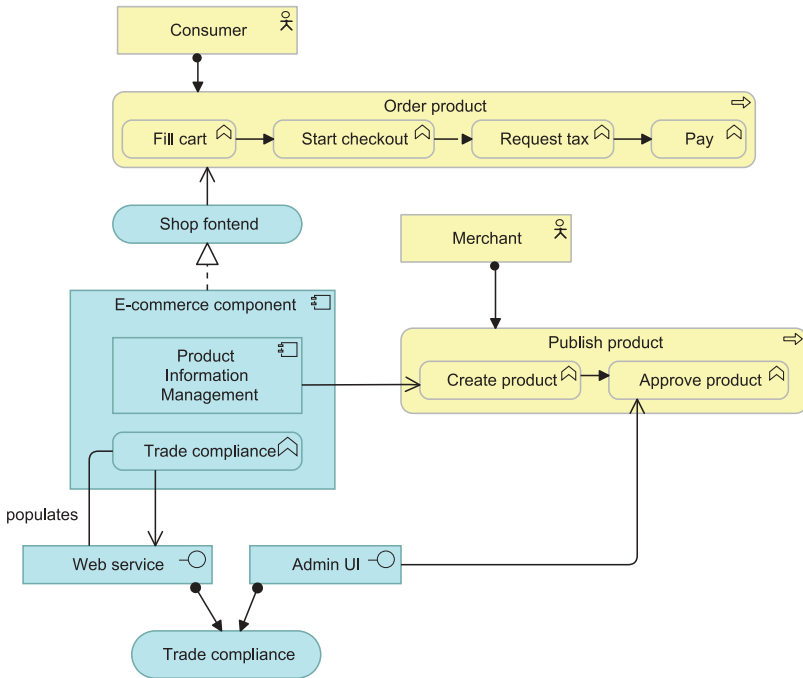


Figure 2. Original reference architecture for the tax compliance service.

country within the fiscal period (Agrawal and Fox 2016). Thus, the service must keep track of all cross-border transactions per country, and apply the correct regulations.

The case study is based on an existing web service that was made available for our research by a global trade compliance consultancy firm. The web service encapsulates the logic required for calculating trade costs, such as foreign VAT and customs, which need to be declared when selling cross-border. To obtain a cost calculation, the corresponding product needs to be published to the service beforehand. The service provides a web application that allows maintaining the product information, and getting an overview of all processed cross-border transactions. The reference architecture proposed by the trade compliance service provider is shown in Figure 2. The model makes use of the ArchiMate modeling notation for enterprise architecture (The Open Group 2016). We apply the notation throughout this work as it allows us to model both business and IT-layer concepts as well as their relationships.

The model illustrates the use of the service within a common e-commerce architecture. The core component of a state-of-the-art e-commerce architecture is a packaged e-commerce solution that encompasses product information management (PIM), customer management, and a storefront. To integrate the trade compliance solution into this architecture, a custom trade cost calculation component is proposed. According to the reference architecture, this is implemented as a customization of the e-commerce package. The customization passes new products onto the service. Furthermore, it carries out the trade cost calculation in real time for incoming orders using the web service.

2.2. IT service adoption

The adoption of IT services often follows the same pattern, consisting of the lifecycle shown in Figure 3. It contains six phases, which occur in nearly every change process that concerns the introduction, update, or replacement of IT services (Aulkemeier, Iacob, and van Hillegersberg

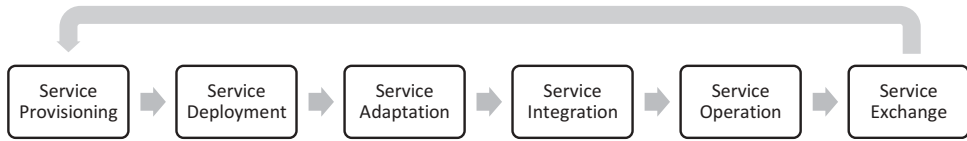


Figure 3. Lifecycle of service adoption and use.

2015). The adoption lifecycle is used as a framework to assess the pluggability of a software component.

Current e-commerce architectures consist of a small set of large systems. More precisely, in most cases, they contain a shop front-end, an enterprise resource planning system for back-office tasks, and a warehouse management system for fulfillment logistics (Aulkemeier et al. 2016a). Each of these application systems covers a large amount of functionality. Such monolithic systems are closely tied to the operational processes of the companies owning them, which makes it hard to detach them (Mandal and Gunasekaran 2003). Furthermore, those systems operate as a whole, thus consisting of tightly coupled modules that are difficult to replace with atomic pieces of functionality encapsulated in other services (Gattiker and Goodhue 2004).

In the following, we assess the reference architecture for the trade compliance service with regard to the six phases. Subsequently, we discuss the shortcomings in the pluggability of current IT services, which emerge from the required resources during each phase of service adoption. The discussion paves the way for an innovative design that embraces these factors.

2.2.1. Service provisioning

During the service provisioning phase, potential services are evaluated and compared. Instruments exist to analyze the fits and misfits of different software packages (Wu, Shin, and Heng 2007). Accordingly, the assessment of software packages can be divided into multiple subphases and can take a year or more. In general, the more complex the system and the higher its impact on the business, the higher the risk and the more critical the service provisioning becomes. On the other hand, finer-grained software components have less impact and, thus, entail less risk, thereby making the provisioning more straightforward. Furthermore, the amount of available information on the software component can be considered as a crucial factor during service provisioning. The existing trade compliance service is not publicly documented beyond the high-level functional outline, and is not available in a service directory, marketplace, or plugin directory of an existing platform. Access to the service and its detailed functionality are subject to individual inquiry with the service provider. Thus, an unbiased assessment and comparison of the service with similar offers on the market require an in-depth investigation. A generic demonstrator for the service is available on-demand, but it is not customer specific. The service does not allow for self-subscription. Technically, there is no straightforward way to assess the service in the target environment. It requires the engineering of a proof-of-concept, including deployment, configuration, and integration.

2.2.2. Service deployment

Software deployment includes activities between the acquisition and execution of a software product and can be a complex and long task, especially in the case of a distributed system (Dearle 2007). The trade compliance service and the two interfaces are hosted by the service provider and do not require allocation of hardware resources, or installation. However, the trade compliance component is required to make the solution work. That component is not issued by the service provider. Therefore, a custom-developed extension of the e-commerce component is required to make the solution work. Such developments also require testing.

2.2.3. Service adaptation

In terms of service adaptation, we can distinguish between configuration and customization. While the configuration allows a user to set up the system according to his/her needs, a customization requires changes in the source code or scripting. In general, the ability of the system to cover a maximum number of business scenarios through configuration decreases the need for customization (Sun et al. 2008). This, in turn, increases the ease of adaptation. Furthermore, a SaaS solution should aim for a high degree of configurability, because customizations are difficult to realize by the user of an external service. The trade compliance service can be configured through the administration user interface. Furthermore, the service must implement the trade compliance component within the e-commerce component, which allows the user to design and adopt the component according to his/her functional requirements.

2.2.4. Service integration

The trade compliance web service consists of two interfaces, one for product registration and another for customs and tax calculation. The product registration interface is used to pass product information onto the service, which will be used for the determination of product-related duties. Users of the trade compliance service must implement the integration between the e-commerce platform (or any other system where product master data is maintained) by themselves. Furthermore, the maintenance and monitoring of that integration artifact must be maintained and monitored by the service user. The same applies for the second web service interface, which is used to initiate the customs and tax calculation during a transaction.

2.2.5. Service operation

The service operation phase encompasses all of the long-term activities that the use of software requires. These can be a major cost factor, as they require permanent human resources in a traditional, non-cloud setting. These activities are multilevel support, including a service desk, and the maintenance of a knowledge base. Furthermore, maintenance of the system is required to improve the service in terms of problem-solving and new features concerning functional or security issues. Major updates might be necessary at the end of the support cycle for a software release, and can have a high impact and require repetitive testing, as well as reengineering of customizations. In the case of the trade compliance service, issues related to service operation occur mainly for the custom-developed component.

2.2.6. Service exchange

At the end of the service adoption lifecycle, the discontinuation or replacement of a service takes place. A critical factor during this phase is whether other services rely on the data or functionality of the retired service. In such cases, data migration, reengineering of other services, and retesting are potential necessities. In the case of the trade compliance service, the trade compliance component within the e-commerce component should be removed. This includes the repeated testing of the e-commerce component. The reference architecture does not contain any dependencies with other services. Thus, canceling the service subscription will not lead to any other side effects.

2.3. Pluggability

From the brief analysis of the IT service adoption lifecycle in Section 2.2, using the example of the trade compliance service, we can conclude that the adoption of specialized services for e-commerce is a resource-intensive endeavor in terms of time, expertise, and cost. Hence, such services can be characterized as having a low pluggability. As a consequence, online retailers that want to adopt new features in order to keep pace with competitors have to accept high investments or wait until their software vendor introduces a similar feature in a future release. In the following, we elaborate on the concept of pluggability. Improving the pluggability of a service is the main design objective of this work.

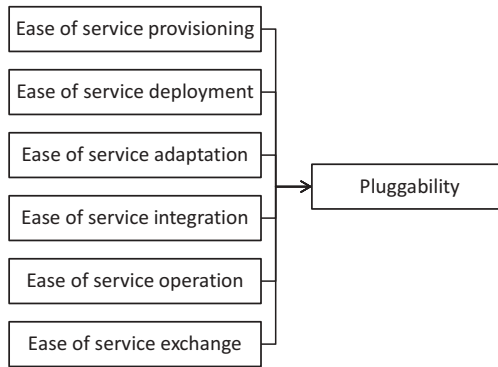


Figure 4. Quality model of pluggability (Aulkemeier, Iacob, and van Hillegersberg 2015).

The concept of pluggability reflects the software quality criteria that drive the ease of adopting new software services (Aulkemeier, Iacob, and van Hillegersberg 2015). It is defined as a quality characteristic to assess software services from a user perspective. It entails factors that have an impact on the need for technical skills and labor to consume the service. Improving the pluggability of a service means giving the service user the possibility to adopt, upgrade, and exchange services with little effort. The goal is to achieve a flexible and fast adoption of services in response to market changes and with the aim of increasing the customer service levels by adopting the latest innovation. Figure 4 shows the quality model with the six criteria that positively impact the pluggability of a software component and that are based on the lifecycle of service adoption.

SaaS solutions, in general, help with the adoption of a service (Benlian and Hess 2011). However, the issues in service integration, operation, and exchange cannot be solved just by switching from an on-premises application to a cloud equivalent. We argue that an improvement of service pluggability cannot be achieved by optimizing just the service itself. In fact, a suitable architecture is also required, which allows service providers to build services that can be consumed in a more pluggable way. Heck and Vervest (2007) argue that a superordinate platform helps improve the capability of services to quickly connect and act as a unit.

3. Toward a pluggable service platform for e-commerce

In the previous section, we described the distinct phases of service adoption and pointed out the low pluggability of IT services for e-commerce in current approaches. In this section, we advance an architecture to increase the pluggability of individual services and the overall flexibility of the IT landscape.

3.1. Platforms for e-commerce

The basic idea of a platform is the separation of stable and evolving components in a system. The platform plays a central role in enabling the variability and evolvability of the system (Baldwin and Woodard 2009). That role can be *to facilitate the exchange* between partners as in a marketplace, to *enable transactions* such as payments, or to act as a technical standard as *software or hardware platforms* do (Sriram et al. 2014). A variety of service platforms exists that cover one or several of these roles. Operating systems, such as Microsoft Windows or Google's Android, are software platforms that provide access to computing resources and can be reused across applications. Modern operating systems also take the role of marketplaces through their integrated app stores (Kimble 2010). By comparison, e-commerce platforms often do not bundle and expose resources but act as intermediaries only, such as business-to-business (B2B) marketplaces. Holzmueller and

Schlüchter (2002) state that the benefits of such B2B marketplaces, which act only as a coordination hub for service providers and users, do not live up to expectations in terms of reduced transaction costs. Software packages are another form of e-commerce platform that enable the web to act as a sales channel. Such platforms are mostly operated by the selling party. SaaS offerings for online stores or business-to-consumer marketplaces, such as eBay, serve the same purpose. Aulkemeier et al. (2016b) explored the architecture of online retail systems and found that such e-commerce platforms cover only a small part of the required functionality to operate an e-commerce channel. More precisely, they provide limited capabilities in terms of warehousing, finance, or customer support, thus assuming the integration with other application components that provide the missing functionality. The realization of such integration is still cumbersome, and requires the engineering and operation of dedicated integration artifacts (Aulkemeier et al. 2016a).

3.2. The service platform model

The architectural pattern we propose in order to ease the adoption of services is the two-sided service platform (Heck and Vervest 2007). Two-sided platforms support service providers and service users at the same time. However, in many cases, such platforms are marginally perceived by the users. They act as invisible engines enabling their clients to provide services (Evans, Hagiu, and Schmalensee 2008). The model in Figure 5 illustrates the actors and relations in a two-sided platform ecosystem (Boley and Chang 2007; Chang and West 2006).

A service provider is a client (or partner) of the platform provider (or platform sponsor) and offers services to the users. In this work, we investigate the e-commerce services offered to retailers. Hence, we consider the e-commerce service provider as the client and the retailer as the user of the e-commerce platform. The e-commerce service makes use of the platform services that are implemented by the platform provider. The platform provider is responsible for implementing long-term stable components of the system that will be used across e-commerce services. In this way, the scattering of functionality and data can be limited. Eventually, the service implementation and evolution, adoption, and cross-service collaboration get simplified.

In other domains, IT services rely on a shared back-end that encapsulates the common components across all services. Platforms following the described principles have been widely adopted in many domains, such as mobile computing, navigation, or gaming (Wareham, Fox, and Cano Giner 2014). Especially the mobile industry applies this two-sided platform approach extensively (Basole and Karla 2011). Developers of mobile applications can rely on several platforms for marketplaces, gaming back-ends, user profiles, advertising, and geo-services. These platforms help service providers reduce implementation efforts, increase the collaboration between services, and enhance the end-user experience. As such, platforms also provide access to shared resources and the replication

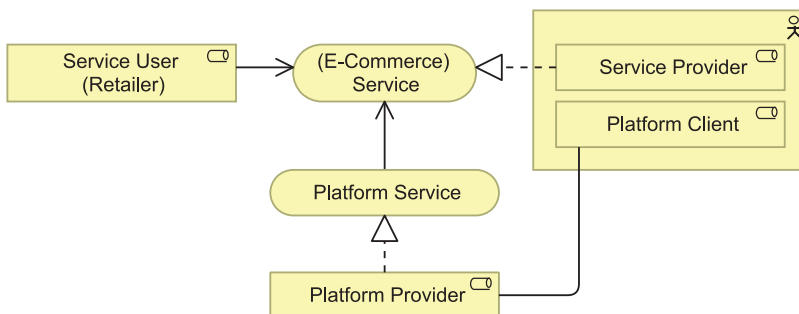


Figure 5. Roles and actors in a two-sided service platform setting.

of data is often not necessary, thus decreasing scattering of data and increasing data quality (Loser, Legner, and Gizanis 2004). Furthermore, the platform design enables service providers to reuse common components and concentrate on the core functionality, allowing them to offer finer-grained services.

To understand the features of the platform architecture, it is beneficial to illustrate how prevailing enterprise application systems are built. According to Fowler (2002), such systems typically consist of several independently deployed layers, usually including a database, an application back-end, and an application front-end (Figure 6). Each individual system encapsulates the data that are required for the provided functionality. Thus, the e-commerce company ends up with many functionally distinct systems with scattered information. The need for interoperability between applications is ignored during their implementation and must be solved later by the user.

Considering the fact that e-commerce services share common data and business logic, the e-commerce platform we introduce (Figure 7) encapsulates a common data model and business rules for e-commerce (Hohpe and Woolf 2003). Furthermore, it allows the definition of rules to let e-commerce companies control the access of diverse services to their resources on an entity level. Finally, the application model extensions permit individual services to extend the data model. This helps keep the common data model simple, yet versatile.

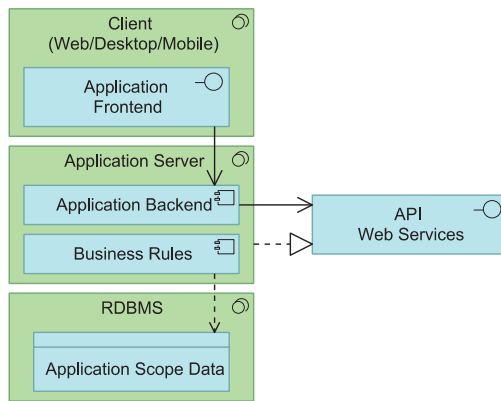


Figure 6. Common application architecture.

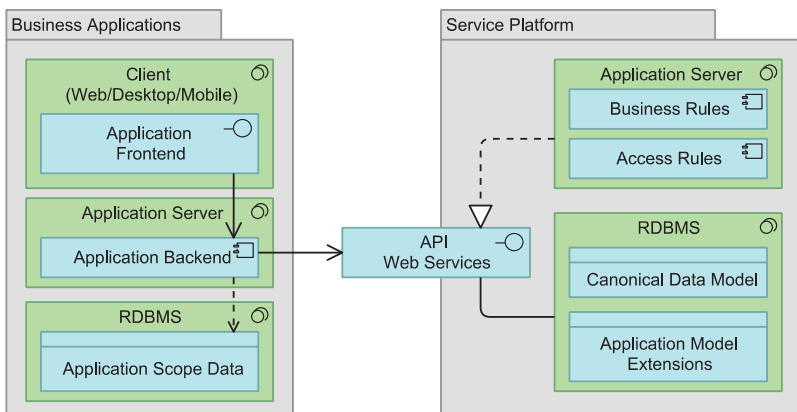


Figure 7. Segregation of service-specific and common resources.

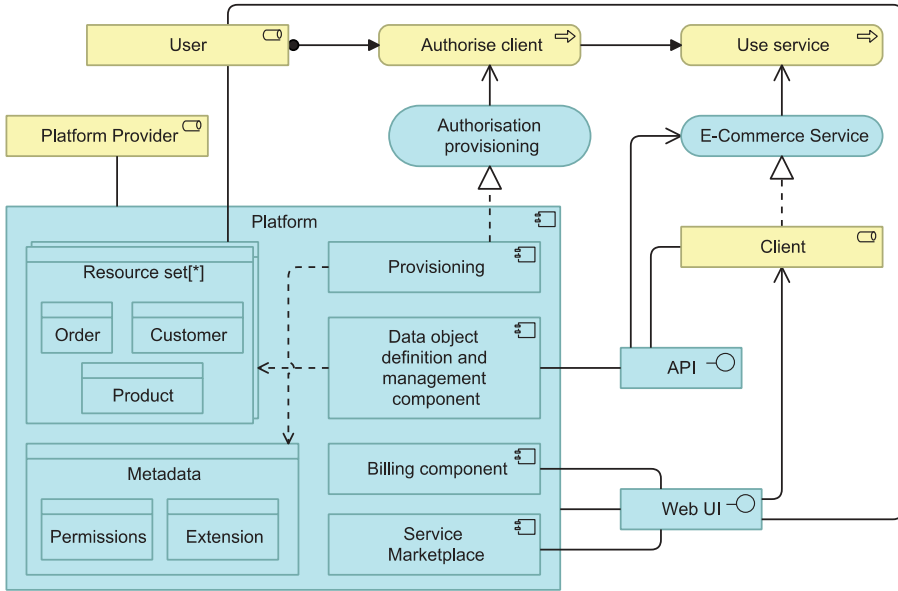


Figure 8. E-commerce platform architecture.

3.3. Platform architecture

Figure 8 shows the architectural model, including the roles, platform components, and processes. The architecture includes three roles, namely the platform provider, the platform user, and the client who offers a service through the platform. There is no preference as to which actors should take which role. It is thus possible that an e-commerce service provider is also operating the platform.

The multiple application components of the platform rely on two types of data objects. On the one hand, a resource set data object is assigned to each user of the platform. It contains the core functional data for e-commerce, such as orders, customer data, and product information. On the other hand, the metadata contains extensions to the core data, as well as the permission that allows access to the various clients.

The platform components accommodate functionality to look up services in a marketplace, and to handle the payment for services. Furthermore, the client can look up and manipulate the functional data, as well as the setup of the metadata required for its service. Finally, the provisioning component manages access to the resources, as later described in detail (Section 4.2).

The platform provides two interfaces: a web application that is used to access the marketplace and a billing component. Furthermore, the core data can be exposed through the web application to facilitate administration tasks on the core data. The API is used by the client to authorize access and to access the core data.

4. A platform-based pluggable trade compliance service

In this section, we describe the implementation of a prototype based on the described platform architecture. Its purpose is to gain insights into the feasibility of an e-commerce service, based on the platform, and to assess its pluggability. To compare the platform approach with the common e-commerce architecture, we reconsider the trade compliance service that has been introduced previously.

4.1. Solution design

Figure 9 shows the architectural model of the platform-based solution for the pluggable trade compliance service. This solution dissolves the monolithic reference architecture and distributes the business logic across different pluggable services. It contains a service for the online shop that handles the sales transactions. A PIM service allows the administration of products. Finally, the trade compliance service automatically passes on new products to the web service, and triggers the computation of cross-border trade costs in real time.

To adopt the numerous services in a pluggable way, services are implemented as clients of the platform. More precisely, domain data, such as product information, is included within the resource set of the platform user, and shared across the services. There is no requirement for using specific services for PIM or the online store as long as they rely on the platform.

The solution supports two process flows that make use of the two interfaces of the trade compliance service:

- The *publish product* flow handles the maintenance of product master data. The user only utilizes the PIM client application to maintain product data. The trade compliance service subscribes to subsequent changes in the product data on the resource set. To do so, the platform provides a special type of API that is able to stream data. This technique allows clients to act upon business events in real time. In our case, a new product in the platform

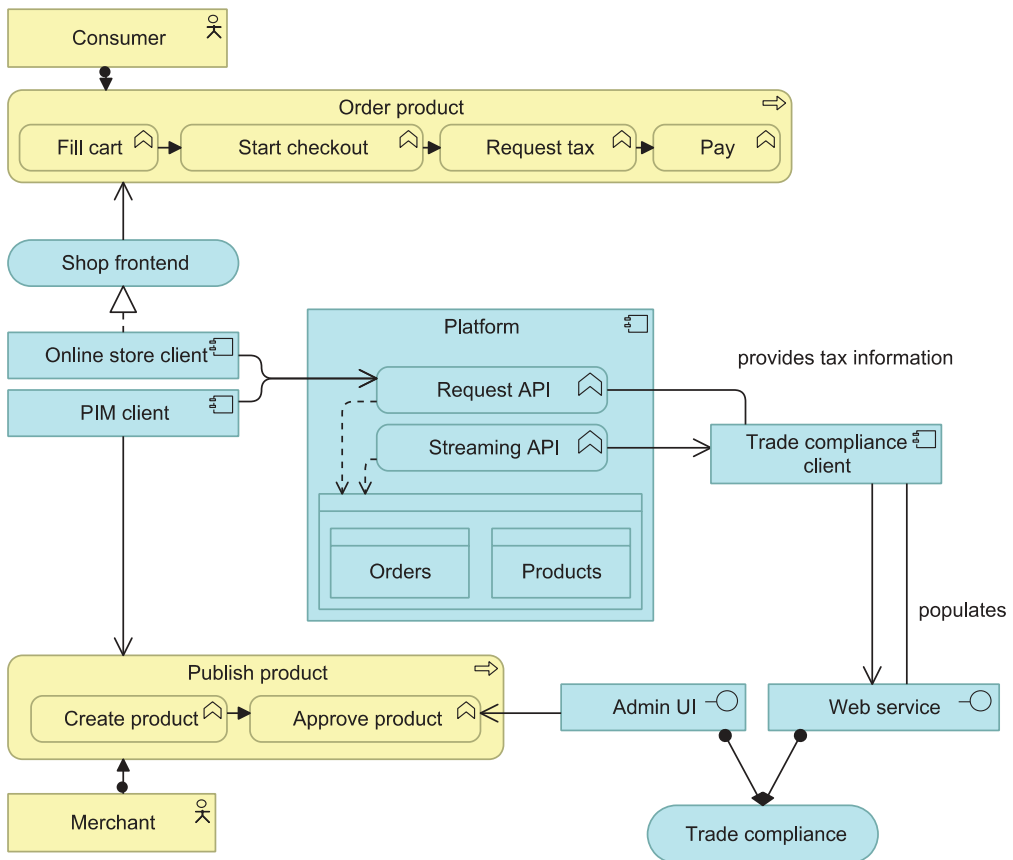


Figure 9. Proposed architecture.

causes the trade compliance client to pick up the product information and publish changes to the trade compliance service through its web interface.

- The *order products* flow makes use of the same streaming technique. The goal is to allow foreign customers to place orders that get updated in real time with the cross-border transaction cost. The crucial part of the implementation is that the additional costs need to get added to the order during the checkout procedure. The trade compliance component picks up the initial order creation event and publishes the additional costs as shown in [Figure 1](#). Therefore, the online store client must reflect changes that have been done after order creation. For that purpose, the platform provides a mechanism that enforces individual services to reflect the public changes of other services.

4.2. Service implementation

For the purpose of resource authorization, the platform implements the OAuth protocol (Hardt 2012). The protocol is frequently used for social media to share a common login across services. Popular services, such as Twitter, Facebook, or LinkedIn, provide the login that other services can reuse. Once the user chooses to subscribe to a new service, he/she is redirected to a page on the platform that provides the client information and the required scope of resource access. Subsequently, the user is presented with a screen, as shown in [Figure 10](#), and can authorize the resource access to the trade compliance service. After confirmation, the user is redirected to the service, and the service is provided with the means to access the user's data.

While the OAuth was originally designed for social media, we argue that the mechanism in enterprise collaboration is the same. The only difference is the amount of the accessed resources. While in social media a limited number of resources, such as username and e-mail, are shared with the third-party service, the e-commerce platform has a larger amount of resources (such as order, products, categories, customers, and returns) that can be shared. Furthermore, a distinction between read and write access to these entities must be made. However, it is possible to reuse the available software for the implementation.

The second challenge, in addition to the delegation of resource access, resides in the implementation of services based on the platform. The new model requires the service to access the data from the platform, instead of a local database. Existing technologies, such as application development frameworks, are not designed with this paradigm in mind. However, with the current trend of cloud platforms in social media and mobile computing, an increasing number of solutions exist that facilitate access to resources through web interfaces.

A potential drawback of the platform architecture is the impact on client performance. More precisely, the access to the centralized data on the platform might cause delays on the client side as the information must be transferred over multiple nodes. Nevertheless, the latest innovations in the

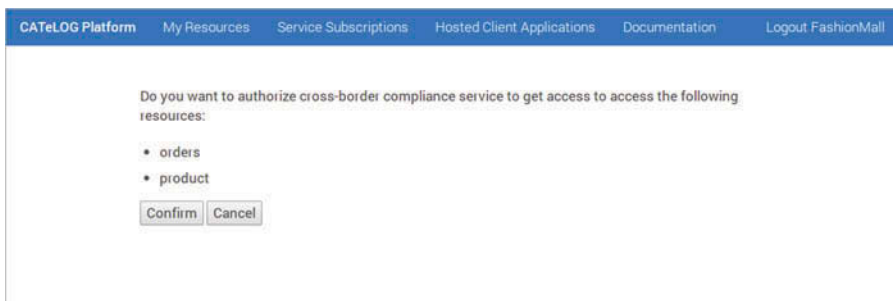


Figure 10. Redirect to platform endpoint for authorization of resource access to the client.

field of application development stress the importance of rich clients that leverage caching mechanisms and asynchronous communication with servers, thus minimizing the impact for the user. In the prototype, we do not address these technologies. However, we think it is important to consider this issue in the future.

Finally, it should be mentioned that the architecture results in a different setting with regard to potential points of failure. The distributed nature of the system assumes many distinct services that contribute to the overall processes. As the outage of a singular node may affect the execution of processes, the overall risk for such incidents increases. Additionally, the platform creates a single point of failure, which also bears a risk for major outages.

The architectural models in [Figures 2](#) and [9](#) illustrate that, from a user's perspective, both architectures share the same business layer. However, the benefit of the platform solution resides in the adoption phase of the individual services. The use of the platform and platform-based services by the retailer offers potential benefits in terms of pluggability and, thus, a shorter time to market. In the next section, we elaborate on these benefits by evaluating the platform regarding the pluggability criteria.

4.3. Evaluation

In this section, we present the evaluation of the cross-border compliance service with regard to its pluggability. The summative evaluation of the design, based on its instantiation, is usually the last phase in the DSRM cycle. However, as Venable, Pries-Heje, and Baskerville (2016) state, episodes of formative evaluation can occur throughout the design process. The design of the architecture was carried out with the criteria for pluggability in mind, and addresses shortcomings in previous design cycles. Furthermore, the summative evaluation presented in the remainder of this section reflects both artificial and naturalistic elements. Naturalistic evaluation entails more complexity as it embeds the design's instantiation in a real-world setting. Our design artifact of a trade compliance service encapsulates the real-world service that was provided by the trade compliance consultancy firm. However, the broader purpose of the design artifact is not the provisioning of a functional service, but its contribution to knowledge. More precisely, the capability of the platform approach toward the pluggability of services should become tangible. Thus, our evaluation is still artificial to a large extent and relies on the critical assessment of the artifact in a controlled environment. Therefore, we rely on the pluggability framework introduced in [Section 2.2](#), which is based on the lifecycle of service adoption and use. Each pluggability criterion is discussed individually, while referring to the corresponding service lifecycle phase.

4.3.1. Ease of service provisioning (EOP)

The ease of service provisioning (EOP) depends on how much the service provider supports the goal of making the capabilities of the solution transparent and allows the user to assess the fit of the solution to the business needs. The existing tax compliance offering is only available upon request and requires individual contact with the sales department of the service provider. It is not possible to discover or learn details about the service unless a business relationship with the service provider is established. Detailed information on the service capabilities is not available. The EOP of the current tax compliance solution is, therefore, on the low end. The advantage of the cloud solution is that it allows the service provider to issue a demonstration environment for customers to get hands-on experience and to assess the service. The trade compliance service does not require an investment in the beginning for licensing and deployment. As the cost for the service arises only from the subscription fee, the investment is distributed evenly over the time of use. Furthermore, the price of a cloud service is more predictable as it will not generate unexpected costs for product implementation and deployment.

The platform architecture aims to facilitate the provisioning of finer-grained services compared to the current service offerings. The trade compliance service adds a very specific functionality to the landscape, which has a very limited impact on the overall system. Thus, there is no need for a long cycle of product assessment that a more comprehensive service would require. The user can experiment with the service to investigate fits and misfits. Compared to the reference architecture of the original trade compliance service (Figure 2), this does not require laborious customizations to the online store. Furthermore, the marketplace component of the platform provides structured information on the service and its pricing, thereby making it easier for the potential user to evaluate its adoption. We can conclude that platform-based solution has major advantages for the user during the provisioning phase and thus a high EOP.

4.3.2. Ease of service deployment (EOD)

As previously mentioned, the ease of deployment (EOD) is a distinctive feature of cloud services in general. The trade compliance service can be used without the need for additional hardware or software artifacts. The platform facilitates subscription to the service by implementing the OAuth protocol. The client has specified the required resource access in advance, during service implementation. As a consequence, the deployment of the service happens in three steps: the subscription to the service, the authentication with the platform, and the authorization of the resource access. The next task of the user is to configure the service according to his/her needs which can already be considered as service adaptation. The EOD can, therefore, be considered as high. In comparison, the original reference architecture requires the user to implement the required tax compliance component, including its technical testing, which leads to a low EOD.

4.3.3. Ease of service adaptation (EOA)

Functional setup of the service and functional testing are the main tasks during service adaptation. The support of the platform-based solution does not go beyond what the original reference architecture offers. The administration interface of the tax compliance service allows the user to configure the service to his needs. However, the service shares a common drawback with other cloud services. The user will not be able to customize the system beyond the functionality anticipated by the service implementer. Therefore, both solutions have a medium ease of adaptation. If the service provider fails or denies implementing certain features, the user will not be able to customize the service beyond the provided functionality, even if he/she has the technical expertise to do so. However, to satisfy customer needs, cloud service providers are forced to improve configurability. The platform can be a means of collecting change request from the various users. Furthermore, the advantage of the platform architecture, in contrast to self-contained SaaS solutions, is that multiple services of the same type can be used. For example, a user has the opportunity to maintain the product data with different services because the underlying data is the same. This is comparable to the use of different e-mail clients on different devices. Instead of adapting one service, the user always has the possibility of choosing another, more suitable, service according to current needs.

4.3.4. Ease of service integration (EOI)

The most significant difference between the original and the new architecture is the way services integrate from a user perspective. While both solutions rely on the same web service interface, the reference architecture delegates the task of integration to the service user, resulting in a very low ease of integration (EOI). The platform-based approach aims at a pre-built integration. With the unified data model of the platform, the integration of services is embedded in the architecture. More precisely, we have a separation of data and functionality, with a federated information system (Busse et al. 1999) whereby the integration problem can be considered as implicitly solved and a high EOI is achieved.

4.3.5. Ease of service operation (EOO)

As described in Section 2.2, there are two types of service-operation-related tasks, namely user support and service maintenance. The maintenance of the trade compliance service, including bug fixing and release updates, is managed by the service provider in both the reference and the platform-based architecture. However, the required customization of the e-commerce component must be maintained by the service user, resulting in a lower ease of operation. User support is also provided by the trade compliance service in both cases. However, if the user relies on the platform-based microservices for the PIM and online store, he/she will have to communicate with each service provider individually. This is not a good practice, as IT service management (ITSM) should aim for a single point of contact (Iqbal, Nieves, and Taylor 2007). In fact, we argue that the platform service provider is very suitable to fulfill the role of a first-level support, coordinating tickets between users and clients. Hence, the platform model should be extended beyond the provisioning of the technical infrastructure and should include ITSM-related tasks.

4.3.6. Ease of service exchange (EOE)

In terms of service exchange, ending the service subscription to the trade compliance service has no impact on the platform-based solution. In the reference architecture, the ending of service subscription requires reengineering of the customization to avoid unwanted errors during checkout. Thus, we obtain a higher ease of exchange with the platform-based architecture.

5. Conclusions

In this paper, we advanced a platform architecture to improve the pluggability of e-commerce services. We found that cloud services have an advantage over traditional application components in terms of pluggability, particularly during service deployment and operation. Furthermore, we found that the platform approach brings additional benefits with regard to service pluggability, which are not addressed by extant cloud computing offerings. More precisely, service integration and exchange can be facilitated through the platform. The marketplace feature may provide additional advantages during service provisioning for users that require a structured overview of comparable services (Agrawal et al. 2013).

Evaluation of the design relies on critical assessment of the artifact's capabilities based on the prototype. The trade compliance case has been established in collaboration with a global trade compliance consultancy firm, and the prototype is based on their real-world service. However, the prototype has not been exposed in an uncontrolled organizational context. In future research, more naturalistic types of evaluation, for example in collaboration with e-commerce companies, are desirable. Furthermore, we identified several potential topics for further research around the platform architecture and service implementation, namely, the impacts on service performance and latency due to the additional node, the potential of client side caching to overcome delays, and potential issues of the architecture with regard to service failures. Furthermore, the integration on the client side, i.e., the support of portlets and widgets through the platform, could be another potential addition to the platform. Other nontechnical extensions of the platform to improve the EOO include the first-level support of the services, as well as bug tracking and change request management by the platform provider.

ORCID

Maria-Eugenia Iacob  <http://orcid.org/0000-0002-4004-0117>

References

Agrawal, D. R., and W. F. Fox. 2016. Taxes in an E-commerce generation. *International Tax and Public Finance*. USA: Springer doi:10.1007/s10797-016-9422-3.

- Agrawal, M., G. Hariharan, H. R. Rao, and R. Kishore. 2013. Competition in mediation services: Modeling the role of expertise, satisfaction, and switching costs. *Journal of Organizational Computing and Electronic Commerce* 23 (3):169–99. doi:10.1080/10919392.2013.807711.
- Aulkemeier, F., M.-E. Iacob, and J. van Hillegersberg. 2015. Pluggable SaaS integration: Quality characteristics for cloud based application services. In Enterprise Systems Conference (ES). Basel: IEEE.
- Aulkemeier, F., M. A. Paramartha, M.-E. Iacob, and J. van Hillegersberg. 2016a. A pluggable service platform architecture for E-commerce. *Information Systems and E-Business Management* 14 (3):469–89. doi:10.1007/s10257-015-0291-6.
- Aulkemeier, F., M. Schramm, M.-E. Iacob, and J. van Hillegersberg. 2016b. A service-oriented E-commerce reference architecture. *Journal of Theoretical and Applied Electronic Commerce Research* 11 (1):26–45. doi:10.4067/S0718-18762016000100003.
- Baldwin, C. Y., and C. J. Woodard. 2009. The architecture of platforms: A unified view. In *Platforms, markets and innovation*, ed. A. Gawer, 19–44. Cheltenham UK, and Northampton, MA: Edward Elgar.
- Basole, R. C., and J. Karla. 2011. On the evolution of mobile platform ecosystem structure and strategy. *Business & Information Systems Engineering* 3 (5):313. doi:10.1007/s12599-011-0174-4.
- Benlian, A., and T. Hess. 2011. Opportunities and risks of software-as-a-service: Findings from a survey of IT executives. *Decision Support Systems* 52 (1):232–46. doi:10.1016/j.dss.2011.07.007.
- Boley, H., and E. Chang. 2007. Digital ecosystems: Principles and semantics. In *Digital ecosystems and technologies conference*, 398–403. Cairns.
- Busse, S., R.-D. Kutsche, U. Leser, and H. Weber. 1999. *Federated information systems: concepts, terminology and architectures*. Forschungsberichte Des Fachbereichs Informatik 99–9, Technische Universität Berlin.
- Chang, E., and M. West. 2006. Digital ecosystems A next generation of the collaborative environment. Paper presented at International Conference on Information Integration and Web-based Applications Services, Yogyakarta.
- Dearle, A. 2007. Software deployment, past, present and future. In *Future of software engineering*, 269–84. Washington.
- Dhar, S. 2012. From outsourcing to cloud computing: Evolution of IT services. *Management Research Review* 35 (8):664–75. doi:10.1108/01409171211247677.
- Evans, D. S., A. Hagiu, and R. Schmalensee. 2008. *Invisible engines: How software platforms drive innovation and transform industries*. Cambridge, MA: MIT Press.
- Fowler, M. 2002. *Patterns of enterprise application architecture*. Boston, MA: Addison-Wesley.
- Gattiker, T. F., and D. L. Goodhue. 2004. Understanding the local-level costs and benefits of ERP through organizational information processing theory. *Information & Management* 41 (4):431–43. doi:10.1016/S0378-7206(03)00082-X.
- Giovanoli, C., P. Pulikal, and S. G. Grivas. 2014. E-marketplace for cloud services. In *International conference on cloud computing, GRIDs, and virtualization*, 76–83. Venice.
- Hardt, D. 2012. The OAuth 2.0 authorization framework. Fremont, CA: Technical Standard, Internet Engineering Task Force.
- Heck, E. V., and P. Vervest. 2007. Smart business networks: How the network wins. *Communications of the ACM* 50 (6):28–37. doi:10.1145/1247001.1247002.
- Hohpe, G., and B. Woolf. 2003. *Enterprise integration patterns: Designing, building, and deploying messaging solutions*. Boston, MA: Addison-Wesley.
- Holzmueller, H. H., and J. Schlüchter. 2002. Delphi study about the future of B2B marketplaces in germany. *Electronic Commerce Research and Applications* 1 (1):2–19. doi:10.1016/S1567-4223(02)00003-0.
- Iqbal, M., M. Nieves, and S. Taylor. 2007. *Service strategy*. ITIL. London, UK: TSO (The Stationery Office).
- Kephart, J. O., and D. M. Chess. 2003. The vision of autonomic computing. *Computer* 36 (1):41–50. doi:10.1109/MC.2003.1160055.
- Kimble, K. 2010. App store strategies for service providers. In *International Conference on Intelligence in Next Generation Networks*. Berlin: IEEE.
- Lendle, A., M. Olarreaga, S. Schropp, and P.-L. Vezina. 2012. There goes gravity: How ebay reduces trade costs. SSRN Scholarly Paper, Social Science Research Network. Elsevier.
- Loser, C., C. Legner, and D. Gizanis. 2004. Master data management for collaborative service processes. Paper presented at International Conference on Service Systems and Service Management, Beijing.
- Mandal, P., and A. Gunasekaran. 2003. Issues in implementing ERP: A case study. *European Journal of Operational Research* 146 (2):274–83. doi:10.1016/S0377-2217(02)00549-0.
- Martens, B., M. Walterbusch, and F. Teuteberg. 2012. Costing of cloud computing services: A total cost of ownership approach. In *Hawaii International Conference on System Science (HICSS)*, 1563–72. Maui.
- Maximilien, E. M., A. Ranabahu, and K. Gomadam. 2008. An online platform for web APIs and service mashups. *IEEE Internet Computing* 12 (5):32–43. doi:10.1109/MIC.2008.92.
- The Open Group. 2016. *ArchiMate 3.0 specification*. Reading: The Open Group.
- Paraiso, F., N. Haderer, P. Merle, R. Rouvoy, and L. Seinturier. 2012. A federated multi-cloud PaaS infrastructure. In *International conference on cloud computing*, 392–99. Honolulu.

- Pedrinaci, C., D. Liu, M. Maleshkova, D. Lambert, J. Kopecky, and J. Domingue. 2010. IServe: A linked services publishing platform. In *Ontology repositories and editors for the semantic web workshop at the extended semantic web conference*, Heraklion. Aachen: CEUR Workshop Proceedings.
- Peppers, K., T. Tuunanen, M. A. Rothenberger, and S. Chatterjee. 2007. A design science research methodology for information systems research. *Journal of Management Information Systems* 24 (3):45–77. doi:10.2753/MIS0742-122240302.
- Sriram, S., P. Manchanda, M. E. Bravo, J. Chu, M. Liye, M. Song, S. Shriver, and U. Subramanian. 2014. Platforms: A multiplicity of research opportunities. *Marketing Letters* 26 (2):141–52. doi:10.1007/s11002-014-9314-1.
- Sun, W., X. Zhang, C. J. Guo, P. Sun, and H. Su. 2008. Software as a service: Configuration and customization perspectives. In *IEEE congress on services*, 18–25. Beijing.
- Venable, J., J. Pries-Heje, and R. Baskerville. 2016. FEDS: A framework for evaluation in design science research. *European Journal of Information Systems* 25 (1):77–89. doi:10.1057/ejis.2014.36.
- Wareham, J., P. B. Fox, and J. L. Cano Giner. 2014. Technology ecosystem governance. *Organization Science* 25 (4):1195–215. doi:10.1287/orsc.2014.0895.
- Weerawarana, S., F. Curbera, F. Leymann, T. Storey, and D. F. Ferguson. 2005. *Web services platform architecture: SOAP, WSDL, WS-policy, WS-addressing, WS-BPEL, WS-reliable messaging and more*. Upper Saddle River, NJ: Prentice Hall.
- Wu, J.-H., -S.-S. Shin, and M. S. H. Heng. 2007. A methodology for ERP misfit analysis. *Information & Management* 44 (8):666–80. doi:10.1016/j.im.2007.09.001.
- Yoon, C., and S. Kim. 2009. Developing the causal model of online store success. *Journal of Organizational Computing and Electronic Commerce* 19 (4):265–84. doi:10.1080/10919390903262644.

Notes on contributors

Fabian Aulkemeier recently received a Ph.D. degree from the University of Twente for his research on platform architectures and service pluggability. During the research project, he collaborated with a consortium of e-commerce companies to deliver components that implement innovative algorithms and decision support processes. Dr. Aulkemeier's research and teaching focus on enterprise architectures, information system design, and software engineering. Previously, he was a consultant for enterprise application customization and integration at Oracle. During that time, he worked for customers across Europe delivering IT solutions in various fields such as SCM, ERP, CRM, and HRM.

Maria-Eugenia Iacob is a Professor in the department of Industrial Engineering and Business Information Systems at the University of Twente, The Netherlands. She holds a Ph.D. degree in Mathematics from the University Babeş-Bolyai of Cluj-Napoca, Romania. She also worked for this university as an assistant professor in the Department of Computer Science. In recent years, her research and courses have focused on enterprise engineering areas, such as enterprise architectures, business modeling and strategic alignment, business process (re)engineering, business rules, supply chain management and integration, service-oriented architectures, model-driven development, industry 4.0, and smart manufacturing. Dr. Iacob is one of the developers of the ArchiMate language for enterprise architecture modeling, and coauthor of the ArchiMate International Standard specification. Her research has primarily emerged as a result of work in multiple projects carried out with industrial partners (e.g., from logistics, healthcare, finance) and is aimed at direct applicability and support for the modern needs of Business-IT-operations technology alignment. Professor Iacob's work has appeared in numerous books and proceedings, as well as journals such as the *Journal of Enterprise Information Systems*, *Information Systems*, *Information Systems Frontiers*, *Information Systems and E-Business Management*, and *Applied Medical Informatics*.

Jos van Hillegersberg is a full Professor of Business Information systems and is Head of the Department of Industrial Engineering and Business Information Systems at the University of Twente. His research deals with innovation of supply chains and business networks using ICT. He is contributing to several national and international projects on the design of collaborative businesses and industrial networks applying ICT such as data analytics, architecture transformation, agent technology, and sensor data. His research has been published in such journals as the *European Journal of Information Systems*, *Decision Support Systems*, *Communications of the ACM*, *Journal of Information Technology*, *Information Systems*, *Journal of Enterprise Information Systems*, *Electronic Markets*, and *International Journal of Production Research*. Dr. van Hillegersberg is chairman of the program committee of the Dutch Research Institute for Advanced Logistics. Before joining the University of Twente, he was on the faculty of the Rotterdam School of Management at the Erasmus University, working on component-based software systems, IT management, global outsourcing, and agent systems for supply chains. Professor van Hillegersberg also worked for several years in business. At AEGON he was component manager for the setup of an internet bank and at IBM he worked on artificial intelligence and expert systems.