Information and Software Technology 55 (2013) 428–441

Contents lists available at SciVerse ScienceDirect

Information and Software Technology

journal homepage: www.elsevier.com/locate/infsof

Constraints for the design of variability-intensive service-oriented reference architectures – An industrial case study

Matthias Galster *, Paris Avgeriou, Dan Tofan

University of Groningen, Department of Mathematics and Computing Science, PO Box 407, 9700 AK Groningen, The Netherlands

ARTICLE INFO

Article history:
Received 21 November 2011
Received in revised form 26 September 2012
Accepted 26 September 2012
Available online 6 October 2012

Keywords:
Variability
Service-oriented architecture
SOA
Reference architectures
e-Government
Case study

ABSTRACT

Context: Service-oriented architecture has become a widely used concept in software industry. However, we currently lack support for designing variability-intensive service-oriented systems. Such systems could be used in different environments, without the need to design them from scratch. To support the design of variability-intensive service-oriented systems, reference architectures that facilitate variability in instantiated service-oriented architectures can help.
Objective: The design of variability-intensive service-oriented reference architectures is subject to specific constraints. Architects need to know these constraints when designing such reference architectures. Our objective is to identify these constraints.
Method: An exploratory case study was performed in the context of local e-government in the Netherlands to study constraints from the perspective of (a) the users of a variability-intensive service-oriented system (municipalities that implement national laws), and (b) the implementing organizations (software vendors). We collected data through interviews with representatives from five organizations, document analyses and expert meetings.
Results: We identified ten constraints (e.g., organizational constraints, integration-related constraints) which affect the process of designing reference architectures for variability-intensive service-oriented systems. Also, we identified how stakeholders are affected by these constraints, and how constraints are specific to the case study domain.
Conclusions: Our results help design variability-intensive service-oriented reference architectures. Furthermore, our results can be used to define processes to design such reference architectures.

1. Introduction

Service-oriented architecture[1] (SOA) has become a widely studied and used concept in software engineering research and practice [1]. SOA is a standard-based and technology-independent distributed computing paradigm for discovering, binding and assembling loosely-coupled software services. Variability on the other hand describes the ability of a software system or artifact to be changed (e.g., extended, customized or configured) for use in a specific context [2]. It specifies parts of the architecture which are not fully defined during early design. Variability is introduced through variation points (i.e., predefined locations in the architecture where change may occur). At these variation points, variants are chosen when instantiating a concrete software system. There are different types of variability, such as variability in features or in business processes. In this paper, we focus on variability in business processes that affect the software architecture. For example, e-government initiatives (involving one national government and many local municipalities) or other head-office organizations, such as banks (one head-office, many branches) need to balance standardization between branches (i.e., offering the same services to all citizens or customers) and variability in the local implementation of business processes.

As SOA aims at aligning business processes and IT, combining variability approaches with service-oriented design would allow us to handle different instances of the same SOA in different organizations and versions. These SOA could be adjusted to operate in diverse environments and within different contexts. However, we currently lack software engineering methods that support designing SOAs that can be adapted in different organizations and for changing situations. So far, variability in service-oriented systems has primarily been addressed at the business process level rather than at the architecture level [3,4]. Also, fundamental design principles of service orientation (e.g., loose coupling, abstraction) do not consider variability as a key issue [5]. Even though these

* Corresponding author.
E-mail addresses: mgalster@ieee.org (M. Galster), paris@cs.rug.nl (P. Avgeriou), d.c.tofan@rug.nl (D. Tofan).

[1] We use singular of the term "service-oriented architecture" when referring to the discipline, technology or paradigm of service-oriented architecture, and plural when referring to more than one service-oriented architecture as an artifact created during software development.

0950-5849/$ - see front matter © 2012 Elsevier B.V. All rights reserved.
http://dx.doi.org/10.1016/j.infsof.2012.09.011

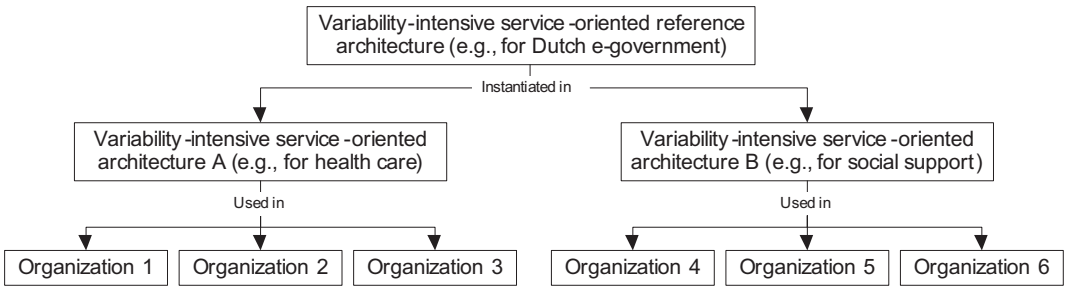

Fig. 1. Example of the context of a variability-intensive service-oriented reference architecture.

design principles have a positive impact on handling variability, it is unclear how variability in SOA can be achieved beyond such high-level and generic principles.

One way to support variability in SOAs is through reference architectures. Reference architectures capture the essence of the architecture of similar systems in an application or technology domain. For implementing variability-intensive SOAs, reference architectures that can be instantiated for different contexts and at the same time support a high degree of variability in instantiated architectures would be useful. For example, in Fig. 1, Organization 1, Organization 2 and Organization 3 use the same architecture to implement health care services, even though their business processes for providing health care services differ. Similarly, Organizations 4, 5 and 6 use the same architecture to implement social support services, even though their business processes for implementing a law for social support differ. This approach requires less development effort and reduces maintenance costs compared to designing new systems for new environments.

However, the design of reference architectures for variability-intensive service-based systems is subject to specific constraints that are currently not well-understood. This is because current service-oriented reference architectures are not built with variability of the instantiated architectures in mind. Furthermore, most current service-oriented reference architectures focus on service publication rather than on holistic architectures that would cover other important constraints on service-based systems [6], e.g., quality of service. Moreover, as most service-oriented reference architectures are developed in academic settings rather than in an industrial context [6], they ignore real-world constraints. Finally, most service-oriented reference architectures are designed based on experience of domain experts, instead of making use of proven and accepted design solutions found in existing architectures, or in other reference architectures [6].

1.1. Why SOA is not enough for handling variability

SOA supports variability through dynamic service retrieval and binding [7]. Furthermore, the right level of service granularity can support or hinder variability. Coarse-grained services are distinguished from fine-grained services that offer less functionality [8]. If services are fine-grained, there are more possibilities to handle variability, which in turn increases the number of service orchestrations and complexity. More details on service granularity can be found in [8,9]. However, several reasons exist why SOA alone is not enough for handling variability:

- Variability occurs in different forms and at different levels of abstraction. For example, variability occurs at the level of individual services, in features and functionality, or in quality attributes (such as performance). However, in SOA, the focus of handling variability is on replacing services, e.g., [10].
- SOA does not support developing applications that fit individual customer needs [11]. Individual services may be reusable, but are not designed to be highly customizable. Service specifications and model are not designed for planned and enforced reuse [11]. Reusing SOAs is difficult because SOA (a) do not treat several similar systems or services as a whole, but rather as multiple separate products, and (b) do not keep generic components in a common base that is evolved and maintained.
- When facilitating variability, we need to consider the integration of services, third party applications, organization-specific systems and legacy systems. The integration of third party applications happens in non-service-based systems as well, but this open-world assumption makes it a more difficult problem for variability [12].
- Dynamic or runtime variability beyond service retrieval and binding must be supported.
- Quality-related issues in SOA have been reported as a top challenge [13]. How to handle variability in quality attributes is still an unsolved problem in SOA.

The problem of variability in service-based systems is related to adaptation of service-based systems. However, adaptation assumes that a system is already running and is then adapted due to changes, either online or offline. Variability on the other hand means that systems are instantiated based on differences of deployment environments. We relate variability to a taxonomy of adaptation of service-based systems introduced by Kazhamiakin et al. [7] with three dimensions:

- Why dimension: Variability includes adaptations to accommodate a particular environment, rather than as corrective, perfective, extending or preventive adaptation. Variability is caused by differences (rather than changes) in environments due to needs of particular customers.
- What dimension: The entity that should be adapted is the business process instance and the context. The adaptation is permanent rather than temporary. With regard to the adaptation aspect, variability as understood in our work is about functional variability.
- How dimension: Variability in service-based systems does not prescribe any adaptation strategy, decision mechanism or adaptation implementation.

1.2. How SOA applications are different from other software

SOA systems differ from other types of software [12]. Thus, they require special treatment for variability. For example, the ownership of services is more critical in service-oriented systems (e.g., for the visibility or interaction of components). In the context of SOA, service providers own services and service consumers merely use an interface through a service contract to consume services. Services as components present the challenge

of meeting requirements for each organization while crossing boundaries between organizations [14]. A higher heterogeneity in customer requirements occurs in service-oriented systems due to anonymous service users [15]. Thus, the range of possible variations between SOA systems might be much broader than in conventional systems and is difficult to anticipate. However, there is no centralized authority that manages variability concerns in the individual parts of the system. Moreover, SOA systems are not developed, integrated, and released in a centrally synchronized way [16]. Services are developed and deployed independently in a networked environment, as well as composed as late as at runtime. Consequently, new methods for coordinating variability are needed (e.g., to decide where, when and how to resolve variability). Technical challenges occur due to the highly distributed nature of service-oriented applications (e.g., security concerns). More details can be found in [12] where the authors discuss differences between "traditional" and service-oriented systems.

1.3. Paper goal and contribution

The goal of this paper is to explore constraints for designing reference architectures for service-oriented systems that require variability in instantiated architectures but cannot be implemented as a product line (for example, because of the application domain). Consequently, concerns of interest are not about variability per se, but about the broader concept of variability-intensive service-oriented reference architectures. Constraints identified in this paper apply to reference architectures for SOA that concern a number of systems, rather than an instantiated service-based architecture for one system. As constraints represent stakeholder concerns, constraints are a first step in the design of reference architectures as the concerns need to be addressed when designing reference architectures. To identify constraints, we present an exploratory case study.

Our work is different from developing high-level and generic reference models for SOA, such as the OASIS reference model [17], the OASIS reference architecture [18] or IBM's foundation architecture [19] in that we focus on variability. However, these architectures might provide high-level guidelines for designing service-oriented systems in general.

1.4. Paper structure

The paper structure and relations between sections are depicted in Fig. 2. We introduce related work in Section 2 and in Section 3 we discuss the industrial context of our case study. Section 4 discusses how we applied case study research. In Section 5 we present the results of our study which are further discussed in Section 6. Section 7 discusses the validity of our study. Section 8 concludes this paper.

2. Background and related work

2.1. Reference architectures

Reference architectures capture the essence of architectures of a collection of similar systems to provide guidance when developing new systems or new versions of similar products. Reference architectures appear in organizations and domains where the multiplicity of applications triggers a need for life-cycle support for all products [20]. Reference architectures are widely used in the software engineering domain. For example, reference architectures for e-contracting [21], security [22,23] or web browsers [24] can be found. Reference architectures are designed by capturing the essentials of existing architectures and by taking into account future needs. A reference architecture can also be built without any previous system or even act as starting point for new systems [25]. However, designers need to be aware of the major constraints, no matter if the reference architecture is created from scratch or based on existing artifacts.

We differentiate product line architectures and reference architectures: Product line architectures tend to be less abstract than reference architectures [26,27], but more abstract than concrete architectures, i.e., product line architectures are one type of reference architecture [28]. In detail, we differentiate product line architecture and reference architecture as follows:

- A product line architecture represents a group of systems that are part of a product line, including processes and infrastructure required in product lines (e.g., core asset development, product development) [29]. A product line architecture is for products that are produced by a single organization. The product line architecture captures the central design of all products of the SPL, including variability and commonalities.
- A reference architecture represent the spectrum of systems in a technology or application domain [24]. It must address business rules, architectural styles, best practices (e.g., decisions, standards). Reference architectures typically do not address variability in a systematic manner. However, in our work we are interested in reference architectures that also support variability.

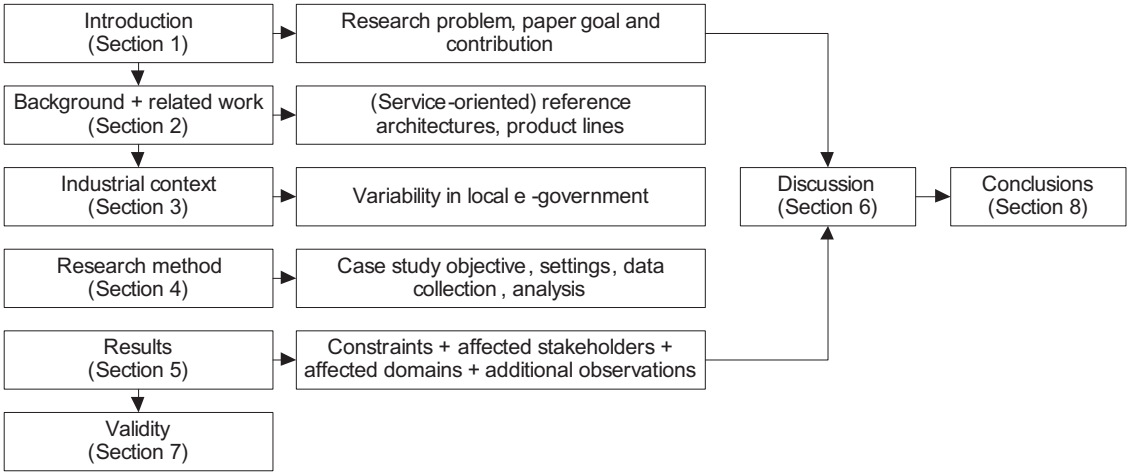

Fig. 2. Paper structure.

We focus on variability-intensive reference architectures rather than on product line architectures due to the following reasons: Variability is a key fact in many systems, not only product lines [30]. For example, when designing service-based systems architects encounter many situations where variability must be handled (e.g., configuration of single systems, customization, multiple deployment/operation/maintenance scenarios, planned evolution of a system over its life cycle, self-adaptive/-healing/-managing systems [30]). Moreover, as variability is pervasive, software engineers need a proper understanding, suitable methods and tools for handling (i.e., representing, managing and reasoning about) variability [31]. Furthermore, product line architectures focus on addressing variability explicitly as "features" and "decisions". On the other hand, in reference architectures variability can be treated as a quality attribute [31]. In this sense, variability is considered in a broader scope and as a concern of different stakeholders, and in turn affects multiple other concerns of the reference architecture.

2.2. Service-oriented reference architectures

Service-oriented reference architectures have been proposed as a special type of reference architectures. For example, the Nexof project developed a high-level service-based reference architecture, based on patterns [32]. Most service-oriented reference architectures have been proposed in the context of government systems [33–35], collaborative work environments [36,37], or e-learning [38,39]. A systematic review on service-oriented reference architectures has been presented by de Oliveira et al. [6]. However, guidelines on how to define reference architectures, and in particular service-oriented reference architectures that support a high degree of variability are missing.

2.3. Service-oriented software product lines

Even though our work is primarily motivated by the lack of variability management in SOA, there has been work on service-oriented software product lines [40]. Service-oriented product lines are becoming a popular approach to manage instances of service-based systems and variability in service-based systems. For example, Lee et al. introduced an approach for developing service-oriented product lines [16]. In a recent technical report, Cohen and Krut discussed managing variation in the context of service-oriented software product lines [41]. Abu-Matar et al. use feature modeling in the context of SOA [42]. Gomaa and Saleh discuss the combination of software product line engineering and web services [43]. Furthermore, Segura et al. propose a taxonomy for variability in web service flows [44]. Our work differs in that we are interested in reference architectures that facilitate the design of variability-intensive service-based applications. Furthermore, our work is complementary to service-oriented product lines as product line architectures could be considered as fine-grained reference architectures for service-oriented product lines. Even though our work does not focus on software product lines, our work still utilizes concepts and definitions from the product line domain.

3. Industrial context

Our research is conducted in the context of local e-government in the Netherlands. In the Netherlands, there are more than 400 municipalities and each municipality serves between 20,000 and 750,000 citizens. To improve the interaction between governments and citizens, municipalities provide a wide range of services through e-government [45]. Thus, laws approved by the national government need to be implemented in local municipalities. However, each of the more than 400 municipalities implements the law autonomously. In this situation, a reference architecture that helps implement the law and at the same time allows municipalities to customize an architecture instantiated from this reference architecture would help. We select this case for our study due to the following reasons:

- In the studied case variability is ubiquitous. This is due to the government structure in the Netherlands that guarantees autonomy to local municipalities. Thus, solutions chosen to implement laws might differ substantially between municipalities.
- The case includes business as well as software-related aspects. It involves processes at different municipalities, as well as changes to these processes over time. Also, it is not only concerned with functionality, but also the satisfaction of quality attributes.
- The case utilizes service-orientation. Software vendors offer software services for supporting laws, in a municipality-independent way [45]. As municipalities in the Netherlands started utilizing the service-oriented paradigm around the year 2004, service-orientation is well-established in this domain.
- In the domain of local e-government, the problem of variability cannot be solved using a product line approach. To comply with e-government regulations, Dutch municipalities must implement SOA-based solutions. However, to serve as many municipalities as possible, SOA have to be customizable. A product line approach however requires the instantiation of concrete architectures by resolving variation points.
- As recently argued by Bouguettaya et al., building e-government systems involves many technical and policy-related challenges [45]. Major issues must be identified and addressed for successful deployment of e-government.

When analyzing the e-government domain, we found two common ways of dealing with SOA in variability-intensive environments. First, each municipality models its own business processes and develops its own systems from scratch. This requires lots of effort. Second, when implementing a new system, an existing system is copied and adapted to local needs. This solution saves time and provides a lot of flexibility. On the other hand, updates (e.g., based on changes to a law) have to be implemented manually for each copy of the system.

4. Research method

As our research is motivated by a practical problem, we apply case study research as an "in-the-wild" method, instead of using "in the lab" methods (e.g., controlled experiments). Also, when investigating variability and reference architectures in an industrial context, we have little control over all involved variables (e.g., people, organizational structures, politics). Moreover, there is only limited access to organizations in regulated industries (i.e., industries which have to comply with federal government regulations, such as local e-government) that utilize SOA. This makes surveys difficult as they would require a broad population of individuals.

Case studies might be descriptive, explanatory, exploratory or evaluatory [46]. We follow the exploratory approach. Exploratory studies are often used in cases where "research looks for patterns, ideas, or hypotheses rather than research that tries to test or confirm hypotheses" [47]. The results of the case study are constraints for the design of reference architectures, rather than a concrete reference architecture or a concrete process for reference architecture design. The research method in the context of the research problem is shown in Fig. 3. The case study design follows the guidelines proposed in [48].

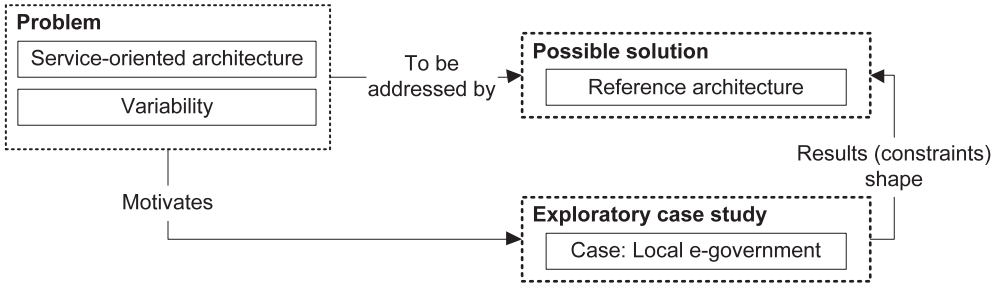

Fig. 3. Research method in the problem context.

4.1. Study objective and research questions

We investigate constraints for variability-intensive service-oriented reference architectures. Constraints are understood as special types of architectural concerns (i.e., any interest of stakeholders in the architecture) and interpreted as problem elements imposed by the environment on the process of designing reference architectures and on the reference architecture itself. This understanding of constraints does not include constraints imposed by the background or experience of architects, but problems imposed by stakeholders, developing organization and the technical environment [49].

4.1.1. RQ1: What constraints exist when designing variability-intensive service-oriented reference architectures?

The design of a reference architecture is complex and there exist potentially many constraints. This is particularly true in regulated environments (such as e-government). Therefore, we aim at identifying what these constraints are for designing variability-intensive service-oriented reference architectures. RQ1 helps us understand constraints that require special attention and that a reference architecture must address.

We study constraints, rather than explore the design of reference architectures. Exploring the design of a reference architecture as a goal would have been too broad. For example, when exploring the design of reference architectures, we would also have needed to study stakeholders, roles, products and artifacts involved in reference architecture design. On the other hand, "constraints" are a subset of issues to consider during the design of reference architectures. Thus, we decided to focus on constraints as constraints are usually factors that make the design of reference architectures more difficult.

4.1.2. RQ2: What constraints are relevant for what stakeholders?

Once we know what constraints occur (from answering RQ1), it also helps design reference architectures to know what constraints are relevant for what stakeholders that are involved in designing and using a reference architecture. Thus, RQ2 aims at characterizing the impact of constraints on stakeholders. This is of paramount importance to the architect who needs to negotiate with stakeholders and make design decisions about satisfying the constraints and the respective stakeholders. We investigate this from the perspective of two major stakeholders: customers (municipalities) and software vendors. When presenting the results to RQ2 in Section 4.1.2 we provide a justification for the focus on these two stakeholders.

4.1.3. RQ3: What constraints are specific to the case study and what constraints are more general?

Once we know the concerns and relevant stakeholders, it is important to investigate whether the constraints are specific to the type of systems we focus on in the case study, or if they are applicable in more general contexts (e.g., for SOA, variability, or regulated industries). For those concerns specific to the case study, architects would need to search for architecture solutions that are mostly domain-specific. For the concerns that are more general (e.g., concerns found in other variability-intensive systems or concerns found in other SOA systems), we should be able to reuse a broader range of architecture solutions (e.g., patterns, tactics or technologies).

4.2. Case study settings

Gerring discusses several case selection strategies, including typical, diverse, extreme, influential, crucial case selection [50]. We use a *single-case* rather a multiple case design [51] as the studied case (see Section 3) is a representative (crucial) and typical case of a variability-intensive environment that utilizes service-orientation. As argued by Yin, representative or typical cases can be studied in single-case designs [51]. Furthermore, our case can be considered revelatory as we had the opportunity to observe and analyze a phenomenon previously inaccessible [51]. These rationales for choosing the single-case design could not be satisfied by a multiple cases [51]. Limitations imposed by the single-case design rather than a multiple case design are discussed in Section 7.

The characteristics of the studied case are (a) it is variability-intensive, (b) the developed software is used by a broad range of customers, (c) it is large-scale and complex (e.g., many components, integration with existing infrastructures and IT systems), (d) the developed systems are enterprise systems rather than consumer software or embedded systems, and (e) it is not part of a product line. Variability between customer organizations is usually too big to provide one implementation for all customers. This means, detailed standardized requirements rarely fit the needs of all customers. On the other hand, developing and maintaining individual solutions for each customer requires significant effort. The unit of analysis of the holistic study design (i.e., one unit of analysis) is reference architecture design, including the management of variability when implementing a law in Dutch municipalities. In particular, we focus on municipalities in a province in the north of the Netherlands.

4.3. Data collection techniques

We collected qualitative data using the following techniques:

- Direct technique: We used semi-structured interviews with eight interviewees [52].
- Indirect techniques: We participated in an expert meeting with twelve experts, and studied technical reports and process descriptions [48].

4.3.1. Interviews

In the interviews, we asked questions to eight subjects (Table 1) from two municipalities and three software companies individually. Each subject had two types of background: SOA expertise

Table 1
Interview participants.

#	Role	Organization	Description
P1	Architect	Company A	Principal business consultant and enterprise architect; familiar with business and architecture aspects of SOA; expert in applying service-oriented computing in municipalities
P2	Architect	Company B	Expert in service-oriented infrastructures, business process modeling and implementation
P3	Manager	Company B	Expert in service-oriented infrastructures and business process modeling and implementation
P4	Program manager	Company B	Expert in systems that implement variable business processes
P5	Advisor	Municipality A	Domain expert in implementing laws in municipalities
P6	Manager	Municipality B	Domain expert in implementing laws in municipalities
P7	Architect	Company C	Domain expert in implementing laws in municipalities
P8	Consultant	Company C	Expert in implementing laws in municipalities, with an emphasis in knowledge management

and expertise in implementing laws, including variability that occurs when implementing laws. Furthermore, subjects had between 5 and 20 years of experience in the problem domain.

Company A is a large global business and technology service company involved in large SOA projects. Company B is a software company with around 500 employees in the Netherlands and India. It is heavily involved in business process modeling and SOA implementation. Company C is a medium-sized consulting company with an emphasis on local e-government. Municipality A is a medium-sized municipality in a northern province in the Netherlands. Municipality B is the largest municipality in a northern province of the Netherlands. Municipalities did not have software architects but still contribute architecturally significant requirements to the design of reference architectures.

We used a semi-structured interview with open questions to allow a broad range of issues to be discussed. Interviews collected information based on the experience of interviewees, rather than based on what they believed is important. Questions were planned but not necessarily always followed in the same order. The semi-structured interviews allowed improvisation and exploration of topics that arose during interviews. Furthermore, semi-structured interviews helped adjust the interview to the focus of expertise and background of interviewees. Interviews were conducted face-to-face. Before each interview, introductory letters were sent to participants. All interviews were recorded and transcribed after the interview. The transcripts resulted in approximately 20 pages per participant. To get feedback from interviewees and avoid misunderstanding, we summarized major findings at the end of the interview. Each interview took 2–3 h.

4.3.2. Expert meeting

An expert meeting was held to collect information about constraints that exist when supporting variability through reference architectures in a service-oriented environment. The full-day expert meeting consisted of formal and informal presentations on SOA, variability, and reference architectures, always focusing on implementing laws in municipalities. Furthermore, the expert meeting included formal and informal discussions. Twelve participants attended the expert meeting (excluding the authors of this paper). Participants of this meeting are listed in Table 2. Participants E1–E5 were industrial experts whereas E6–E12 were academic experts involved in industrial projects. Academic experts were neither members of our research group, nor involved in this research. During the meeting, notes were taken and transcripts were created.

4.3.3. Existing documentation

Existing documentation refers to the analysis of a collection of business processes and their implementation in municipalities [53]. The analyzed documents include interviews with civil servants as well as IT staff in municipalities from seven municipalities in the north of the Netherlands. It describes the current situation of implementing laws in municipalities. Also, the documents discuss requirements for process support with SOA in municipalities. In detail, they describe how processes are currently performed, what similarities and differences can be found between municipalities, and changes to processes over time [53].

4.4. Data analysis

We performed content analysis based on the interview transcripts, transcripts from the expert meeting and excerpts from the description of business processes and their implementation in municipalities. From the predefined questions on the questionnaire used in the interviews, we constructed a set of labels referring to topics that we expected to arise from the data and that related to RQ1, RQ2 and RQ3. During the analysis, this set of labels evolved. All interview transcripts, transcripts from the expert meeting as well as the reports on practices in municipalities were thoroughly read, and phrases of interest were coded with the labels to reflect the topic of that phrase [54]. After the initial coding, we looked at groups of code phrases and merged them into categories. As our data was collected within a case study, the data is context sensitive. Thus, we performed iterative content analysis to make inferences from collected data in its context [55]. We used constant comparison to analyze our data and to generate categories of data [54]. We used a simple method for open coding [56] where one code can be assigned to many pieces of text, and one piece of text can be assigned to more than one code [57].

Analyzing qualitative data required integrating data where different participants might have used terms and concepts with different meanings, or different terms and concepts to express

Table 2
Participants of expert meeting.

Participant	Role
E1	Expert in service-oriented computing with experience in government projects
E2	Representative from a SOA provider for municipalities; domain expert in applying service-oriented computing in local municipalities
E3	Representative from a SOA infrastructure provider; domain expert in service-oriented infrastructures and business process modeling and implementation
E4	Representative from a SOA infrastructure provider; domain expert in service-oriented infrastructures and business process modeling and implementation
E5	Representative from a municipality; domain expert in implementing laws in municipalities
E6–E9	Four experts in distributed systems from academia
E10–E12	Three experts in business processes from academia

the same thing [58]. To address this problem, we used reciprocal translation [58]. Reciprocal translation helps summarize newly identified constraints that relate to other similar constraints by translating similar constraints into one another. Also, we investigated if some constraints were sub-issues of other constraints. For that reason, line of argument synthesis was used to identify the "main theme" of different constraints [58]. Line of argument synthesis could be applied to all constraints in an iterative manner until higher level constraints were not overlapping and no sub-constraint was part of another higher-level constraint. The final results were checked with representatives from Company A as a major contributor to and user of reference architectures.

5. Results

In this section, we discuss the case study results and how they relate to answering the research questions.

5.1. RQ1: What constraints exist when designing variability-intensive service-oriented reference architectures?

Constraints are problem elements imposed by the environment. Consequently, constraints relate to the process of *designing* and *introducing* reference architectures as well as to the reference architecture as the product of the design process. We identified ten constraints, with some constraints grouped into organizational constraints and integration-related constraints (Fig. 4). These groups emerged as "organizational" issues and "integration-related" issues appeared as reoccurring themes in some constraints during data analysis.

5.1.1. Organizational constraints (C1–C3)

Organizational constraints are related to the organization that is to use the reference architecture and SOA to address variability.

5.1.1.1. Maturity of legacy architecture (C1). A mature architecture is an explicitly defined architecture that serves a concrete purpose. It is implemented, maintained and enforced in an organization. It describes the most important stakeholders involved in business processes, responsibilities of components, and facilitates quality management, etc. If no mature architecture exists, designing and introducing a service-based reference architecture is likely to fail.

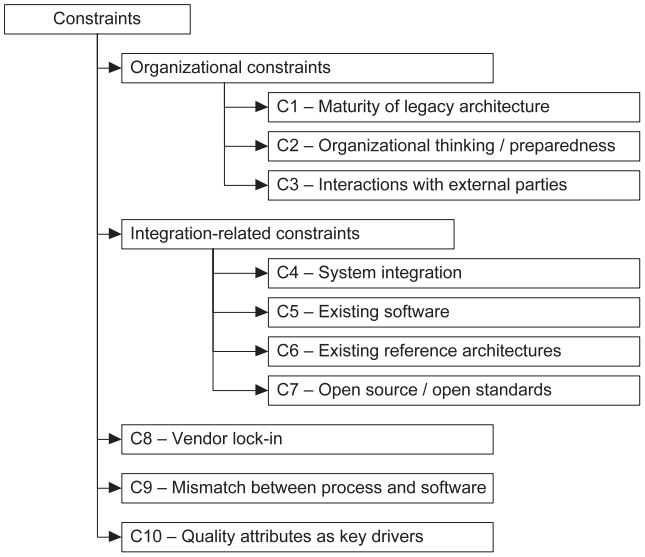

Fig. 4. List of constraints.

A mature legacy architecture is the precondition for *introducing* a service-based reference architecture. Even though the legacy architecture plays a role for introducing and designing any kind of reference architecture, we found it particularly important in the context of service-oriented reference architectures as the legacy architecture might change significantly when migrating to a SOA-based environment. Most importantly, a mature legacy architecture follows principles for "good" design, such as high cohesion, high modularity and low coupling. Following these principles would also ease accommodating variability in a reference architecture.

5.1.1.2. Organizational thinking/preparedness (C2). Organizational thinking is related to organizational issues (e.g., distribution of funding) rather than technical issues in organizations. When a service-based reference architecture is implemented, different departments within an organization need to a) share information with other departments, but also b) get things from other department. This awareness of variability between different departments of organizations and related organizational thinking/preparedness is a precondition for *introducing* a service-based reference architecture. In the example of e-government, changing organizational thinking in employees is often achieved through training that takes place when introducing service-oriented reference architectures. As we show in Section 5.3, this constraint is not so much related to variability as such, but more to the use of a service-oriented reference architecture.

5.1.1.3. Interactions with external parties (C3). For executing business processes, parties outside an organization perform services. For example, a reference architecture for a law to regulate social services for citizens must consider interactions other government authorities (e.g., citizen registry) to get advice, or for receiving medical data from hospitals. In the context of SOA, there are potentially many unknown external parties that a new reference architecture must interact with. These unknown external parties cause variability. This is a constraint on the process of *designing* reference architectures.

5.1.2. Integration-related constraints (C4–C7)

5.1.2.1. System integration (C4). System integration refers to the integration of the reference architecture in existing eco-systems. For example, a new reference architecture for e-government in the Netherlands needs to be integrated into existing software and IT structures in local municipalities. This variability in software and IT structures as well as in eco-systems in which the reference architecture is used needs to be understood when *introducing* the reference architecture in an organization. Even though SOA claims seamless integration through separation of interfaces and implementation of software services, system integration is still a constraint for reference architecture design.

5.1.2.2. Integration of existing software (C5). In addition to integrating the reference architecture in existing systems (C4), there is software that needs to be integrated into (and become part of) the reference architecture. For example, a reference architecture in local municipalities in the Netherlands needs to include systems to manage citizen data. Some of the software that is to be integrated might not be service-oriented, thus requiring wrappers in the reference architecture. This requires an understanding of the existing software when *designing* a reference architecture. With regard to variability, the need to integrate existing systems causes variation points in the architecture that are often unknown at design time of the reference architecture.

5.1.2.3. Integration of existing reference architectures (C6). In a service-oriented environment, other reference architectures could exist. These reference architectures constrain the *design* of new reference architectures as new solutions must fit in these reference architectures. In our case study, two reference architectures exist which partially shape the design of more specific reference architectures: NORA (Dutch reference architecture for the public sector at the national level) which utilizes the SOA paradigm, and GEMMA (reference architecture for Dutch municipalities). NORA applies to Dutch e-government in general whereas GEMMA applies to local e-government and municipalities. Such existing reference architectures are usually rather abstract and one cannot directly derive concrete architectures from them. Instead, new variability-intensive service-oriented reference architectures (e.g., for social security, large municipalities) could be developed. These reference architectures allow us to derive specific architectures. This constraint is indirectly related to variability as existing reference architectures are one way of facilitating variability [59].

5.1.2.4. Integration of open source/open standards (C7). There is a trend towards using and integrating open standards and open source software when *designing* reference architectures. This is because high level reference architectures (see C6) usually prescribe some standards for data exchange (e.g., StUF as the standard data exchange format in GEMMA). Usually, no proprietary standards are used, except for specific adapters if no standard in a reference architecture exists (e.g., company A that participated in the case study has around 30 specific adapters). SOA itself heavily relies on interoperability of services and other parts of the system.

5.1.3. Vendor lock-in (C8)

The existence of variability is one cause for vendor lock-in because vendors possess knowledge on how to deal with variability, and not the municipalities. Vendor lock-in means that customers are restricted in changing their system without the involvement of the vendor, despite the use of open standards. Customers try to reduce vendor lock-in, but this is not always possible, given the small market of software vendors in certain domains and the required expertise. As stated by interview participant P5, applications often cannot communicate without involving vendors as only vendors understand the dataflow between applications and know how the data is processed. This means, software protocols provided by e-government standards are not specific enough to support the transfer of data from one application to another application. Avoiding vendor lock-in affects the *design* as well as the *introduction* of a reference architecture.

5.1.4. Mismatch between software and processes (C9)

As many applications are provided by vendors off-the-shelf, these applications do not always match the processes inside the organizations. This is a particular problem in the context of SOA as SOA integrates business process and solution design. For example, digital application forms for social services in the system are often not the same as the paper version of a form used by civil servants in municipalities. This is because the digital forms are standardized whereas the business processes and the data required to complete a business process in municipalities vary. Moreover, management information that is required in the paper trail often cannot be retrieved from software-based information systems. This confirms the findings from a previous study that the lack of customizability prevents organizations from moving to a service-oriented environment [60]. The mismatch needs to be limited as far as possible while *designing* and *introducing* a reference architecture.

5.1.5. Quality attributes as key drivers (C10)

The key drivers of a software architecture are not features but quality attributes. In the case of variability-intensive service-based systems, several quality attributes are relevant when *designing* reference architectures. We discuss these in Sections 5.2 and 5.3 when we examine how constraints are related to stakeholders and to the case study domain.

5.1.6. Relating constraints to sources and data collection techniques

Table 3 relates constraints to the techniques that we used for data collection (see Section 4.3). This is to indicate the validity of our results by showing that all constraints can be traced to more than one source. A constraint might originate from more than one participant of the interviews or the expert meeting. Table 3 shows that all constraints arose during the interviews. Furthermore, organizational and non-technical constraints tended to arise during the expert meeting but are not found in documentation (C1 and C2).

5.2. RQ2: What constraints are relevant for what stakeholders?

We used two perspectives for stakeholders: "Municipality" and "Vendor" represent the most important stakeholders that a) will be using instantiated architectures as end users (municipalities), and b) will be using the reference architecture to instantiate concrete architectures (vendors). In the context of the case study there are potentially more stakeholders, such as the national government and citizens. However, the needs of these stakeholders are taken into account by municipalities because municipalities formulate their IT needs based on all other stakeholders (e.g., if citizens or the federal government are concerned about security, it becomes a concern of the municipality and the vendor has to find a solution for it). Consequently, the case study was not concerned with how municipalities come up with these needs, and why. Municipalities are treated as customers who demand solutions from software vendors. In Table 4 we map the constraints to perspectives (i.e., stakeholders). The mapping was done based on the labels assigned to constraints during data analysis.

From Table 4 we see that all constraints (except C1) can be related to the customer perspective (municipalities), i.e., C1 is unique for software vendors. This is because an understanding of existing systems and business processes requires technical details that are usually not of interest for customers (municipalities), but they need to be understood by vendors.

Table 3
Constraints and data sources of constraints.

Constraint	Interviews	Expert meeting	Documentation
C1 – Maturity of legacy architecture	✓	✓	
C2 – Organizational thinking/ preparedness	✓	✓	
C3 – Interactions with external parties	✓	✓	✓
C4 – System integration	✓	✓	✓
C5 – Existing software	✓		✓
C6 – Existing reference architectures	✓	✓	
C7 – Open source/open standards	✓	✓	✓
C8 – Vendor lock-in	✓	✓	✓
C9 – Mismatch between process and software	✓	✓	✓
C10 – Quality attributes as key drivers	✓		✓

Table 4
Mapping of constraints to perspectives (stakeholders).

Municipality	Vendor
–	C1 – Maturity of legacy architecture
C2 – Organizational thinking/ preparedness	C2 – Organizational thinking/ preparedness
C3 – Interactions with external parties	–
C4 – System integration	C4 – System integration
C5 – Integration of existing software	C5 – Integration of existing software
C6 – Integration of existing reference architectures	C6 – Integration of existing reference architectures
C7 – Integration of open source/open standards	C7 – Integration of open source/open standards
C8 – Vendor lock-in	–
C9 – Mismatch between software and process	–
C10 – Quality attributes as key drivers	C10 – Quality attributes as key drivers

All constraints (except C3, C8 and C9) can be related to the vendor perspective. C8 is not a constraint for vendors as vendors are the cause for vendor lock-in. C9 (mismatch between software and process) relates to customers as in many cases standard solutions are provided by vendors which might not fit the processes within organizations. Interactions with external parties (C3) is also something that customers are more concerned about, as external parties are usually very specific to the individual municipalities. In the following, we explain each cell of Table 4:

Municipality:

- C2: When a service-based reference architecture is implemented, different units within an organization are affected. For example, introducing SOA changes the way projects are financed (e.g., departments might spend 50% of their funding on a shared SOA project). The organization introducing a service-based reference architecture must be ready for SOA (i.e., willing to share). This requires a different way of thinking within organizations. For example, sharing services that are needed from other parts of the organization might be more important than having a linear process. Also, the IT department of an organization changes after introducing SOA. Rather than having groups for individual applications, there will be groups of applications but also a group for shared applications and infrastructure. Consequently, the maintenance organization will change and workforce needs to be restructured. This might take years to introduce; however, it should be in place before implementing a SOA-based solution (for example, as reported by P1, it took 2 years for a police department in a European country to make this switch).
- C3: Municipalities need to interact with external parties in order to complete certain tasks (e.g., obtaining medical assessments to evaluate the eligibility of citizens for social services).
- C4: System integration is particularly relevant for municipalities. For example, in municipality A, a reference architecture needs to be integrated in an environment with 150 different applications from different vendors. In municipality B, the reference architecture would need to fit into around 400 applications.
- C5: The integration of existing software into the reference architecture is essential for municipalities. In the municipalities that we studied, database management systems, customer management systems, enterprise resource planning systems and many other types of software are used.
- C6: The integration of existing reference architectures is also important for municipalities. Even though GEMMA provides lots of standardization through reference processes, data model standards and data exchange formats, there is still too much variability in data exchange. This causes problems as municipalities cannot interoperate with other municipalities.
- C7: Open standards and open source are heavily encouraged by municipalities. For example, in municipality A, five of the most important applications are open source, e.g., the database server as well as the customer management system.
- C8: Most domain knowledge resides with vendors. For example, vendors have information about legal requirements as well as technical knowledge related to implementing laws. This kind of information is difficult and expensive for municipalities to obtain.
- C9: Given the high degree of variability in municipalities, there is often a mismatch between the process and the solution to implement the process. Vendors only provide out-of-the-box solutions which often do not meet the local requirements.
- C10: Quality attributes requested by municipalities include performance and scalability. From a municipality's perspective, a service-oriented reference architecture must be able to work in a large scale (e.g., local e-government, central government) and integrate with many external systems. In particular, to guarantee high reliability and low downtime, the governance of the instantiated architecture must be supported.

Vendor:

- C1: Checking the maturity of an architecture is a concern for software vendors who need to make sure that existing architectures support the introduction of a SOA-based solution. Often, architecture maturity checks are performed before moving to a SOA-based solution using architecture maturity models [61,62].
- C2: Vendors need to ensure that the customer organization is willing and able to adjust its organizational thinking when introducing a SOA-based solution.
- C4: There is a large degree of variability in systems into which a reference architecture would need to be integrated. Vendors must be aware of all systems to make a reference architecture fit.
- C5: Integration of existing software might require the integration of software from different vendors that are not necessarily service-based. In the small municipality that we studied, only 10% of the applications are service-based.
- C6: Existing reference architectures, such as NORA, do not only provide standardization but are also the starting point for any reference architecture in the context of e-government. For vendors, GEMMA is also important as it drives the integration by providing communication and data models, standardization in processes and protocols.
- C7: Open standards are usually defined in existing reference architectures. Having open standards allows vendors to provide solutions more easily and to communicate with third-party software. For example, open standards allow vendors to exchange information between different parts of a system (e.g., external citizen registries and municipality-specific databases).
- C10: Privacy and security issues reduce the willingness of using cloud computing as part of SOA. As business managers in municipalities are often afraid to lose control over data, the reference architecture must support the qual-

ity attribute *privacy*. This is because municipalities want to be autonomous and separate from other municipalities.

5.3. RQ3: What constraints are specific to the case study and what constraints are more general?

We are investigating variability-intensive, SOA-based systems for local e-government. We have thus selected three "topics" as distinguishing criteria of the type of system we are considering in our case study: "SOA" as the general architectural pattern, "Variability" as we are interested in variability-intensiveness, and "Regulated industry" as the application domain of the case study. To decide what constraints are specific to the case study, we mapped constraints to topics based on the labels assigned to constraints. A constraint might be assigned to a single topic or several topics. Table 5 illustrates the resulting mapping of constraints to topics. In Table 5 some constrains related to variability could also be related to SOA. However, we related constraints to topics based on the primary topic, i.e., the topic that has most influence on them. Constraints that appear for all topics in Table 5 are concerns which are specific to the case study, i.e., C3 and C10.

From Table 5 we see that C1 and C7 can be related only to "SOA". This means that the maturity of legacy architectures and the integration of open standards is a concern not only for variability-intensive service-based systems, but for service-orientation in general. As a mature legacy architecture would also comply to good design principles (C1), it is also related to variability. Furthermore, system integration and the integration of existing software is one of the promised benefits of SOA. The role of open source and open standards (C7) is also highly linked to service orientation; as stated by one of the interview participants, open standards are the major reason for many software vendors to utilize SOA. C2 and C3 apply to SOAs in regulated industries and are not specific to variability aspects. C4, C5 and C9 are constraints specific to variability-intensive systems (not necessarily service-based). Other systems need to be integrated into the reference architecture and the reference architecture has to be integrated with broader ecosystems (C4, C5). Vendors need to be aware of those systems. Only C8 is a constraint that is specific to the case study application domain and variability. C10 is specific to variability-intensive service-oriented reference architectures in regulated environments in the sense that specific quality attributes are key drivers for this type of systems (scalability, interoperability, performance, reliability, privacy, security). Variability is often considered as a quality attribute itself as it plays a paramount importance in this domain. C6 is a constraint not mapped to any of the three topics: The integration of existing reference architectures is not specific to any topic and occurs in the context of reference architectures which are not necessarily in a regulated environment, service-based or variability-intensive.

Table 5
Mapping of constraints to topics.

SOA	C1 – Maturity of legacy architecture C2 – Organizational thinking/preparedness C3 – Interactions with external parties C7 – Integration of open source/open standards C10 – Quality attributes as key drivers
Variability	C1 – Maturity of legacy architecture C3 – Interactions with external parties C4 – System integration C5 – Integration of existing software C8 – Vendor lock-in C9 – Mismatch between software and process C10 – Quality attributes as key drivers
Regulated industry	C2 – Organizational thinking/preparedness C3 – Interactions with external parties C8 – Vendor lock-in C10 – Quality attributes as key drivers

The mapping of constraints to the three topics will be a useful input during the design of the reference architecture itself, as one can look for reusable architectural solutions in the context of the three associated topics. To further analyze the constraints with regard to their generalizability, we related them to general software service engineering challenges [63]. All our constraints, except C6 and C8 were reflected in these challenges.

5.4. Additional observations

Below are additional observations that we made when analyzing the data. These observations are not directly related to answering RQ1 to RQ3.

Observation 1: Open standards and open source software can facilitate rather than inhibit reference architectures. This is a surprising observation as in regulated industries (such as e-government) organizations (such as municipalities) tend to protect systems through closed software and closed standards to keep full control over systems. However, in our case study, closed systems are characterized by two major problems: (a) they cause vendor lock-in, and (b) they inhibit adaptation to changing needs.

Observation 2: There is only a minor overlap between the constraints we identified in this industrial case study and topics of general challenges in service-oriented system engineering [13]. Out of 45 challenges identified by Gu and Lago based on a literature study, only "quality" is the topic which overlaps with one of our types of constraints. This is an indicator that designing SOAs in e-government goes beyond the usual challenges faced in service-oriented systems engineering.

Observation 3: We found that "Regulated industry" (e-government in the Netherlands) has several drivers for variability: First, the size of a municipality (large municipalities have similar processes), second, the economic status (e.g., some municipalities are more wealthy than others), third, the international appeal (e.g., harbor cities), and fourth, the demographic situation (e.g., the amount of senior citizens). These drivers also have implications on how municipalities deal with variability, or how seriously constraints affect municipalities. For example, large municipalities would potentially develop their own software systems and thus benefit less from a reference architecture, compared to smaller municipalities that do not have enough resources to develop systems from scratch. Furthermore, small municipalities suffer more from vendor lock-in (C8) as their existing software is often from one vendor (C5). Wealthy municipalities can usually spend more resources on training employees to ensure organizational thinking and preparedness (C2). Municipalities with international appeal tend to have more involvement of external parties (C3) than municipalities with mostly domestic audience.

6. Discussion

Based on the results discussed in Section 5, we provide a general discussion of our findings. Furthermore, we discuss how the results of the case study can be used to support the design of reference architectures. We also elaborate on the applicability of the results beyond the domain of the case study.

6.1. General discussion of results

Some of our results (e.g., constraints such as C1 – maturity of legacy architecture, C2 – organizational thinking, or C5 – integration of existing software) may also be applicable to other

types of systems and reference architectures than the ones studied in this paper. However, in order to design variability-intensive reference architectures for service-based systems, it is necessary to be aware of the constraints that exist in this domain. Also, it could be argued that the constraints are not new or are common architectural guidelines. However, we believe that it is important to provide evidence about which constraints apply in the given problem domain, rather than assuming constraints. By posing RQ3, we tried to define constraints that are specific to the case study domain. In summary, the fact that constraints occur in the context of SOA does not mean that they do not occur in other contexts.

6.2. Design of variability-intensive service-oriented reference architectures

The results from our study shape variability-intensive service-oriented reference architectures. Furthermore, the constraints affect the process of designing such reference architectures.

6.2.1. New variability-intensive service-oriented reference architectures

In Fig. 5, we show how a new reference architecture would fit into the context of existing reference architectures in the domain of our case study, i.e., Dutch e-government. This figure shows that a new variability-intensive reference architecture for municipalities would refine a reference architecture like GEMMA and provide more details about the implementation of new laws.

In Dutch e-government, new variability-intensive reference architectures would be needed for social security, or for large municipalities. The new reference architecture would need to comply with constraints from Section 5:

- The maturity of the legacy architecture (C1) and organizational thinking/preparedness (C2) are prerequisites for organizations that use the new reference architecture (or architectures instantiated from it).
- These prerequisites can be part of the reference architecture specification.
- Interactions with external parties (C3) can be enabled in new reference architectures by defining generic interfaces in the reference architecture specification.
- The new reference architecture would help integrate existing software and legacy systems (C4 and C5).
- The new reference architecture would integrate existing reference architectures (C6), e.g., GEMMA.
- Open source and open standards would be facilitated, as these are part of existing reference architectures, for example GEMMA (C7). This reduces vendor lock-in (C8). Furthermore, explicitly excluding proprietary systems that cause vendor lock-in can become part of the reference architecture specification.
- Avoiding mismatch between software and process (C9) can be one of the main drivers for the reference architecture. Current processes would need to be analyzed in detail and be expressed in the requirements for a reference architecture (e.g. through a business or domain model).
- The most important quality attributes (scalability, interoperability, performance, reliability, privacy, security) would be supported by architecture decisions and evaluated through an architecture evaluation method. Examples of decisions that can help achievement of quality attribute requirements are a) multi-tenant solutions with different municipalities having their own tenant, and b) open standards (C10).

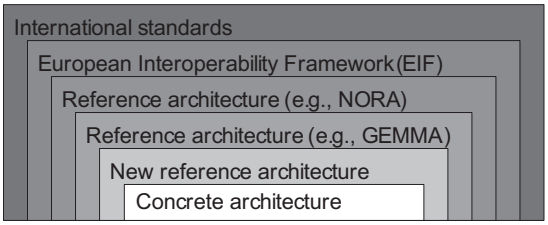

Fig. 5. Context of new reference architectures in the case study domain.

6.2.2. Frameworks for designing variability-intensive service-oriented reference architectures

Cloutier et al. present a high-level model for reference architecture design, covering the collection of information from existing systems and evolving the reference architecture based on standards, emerging technology, changing stakeholder needs and maturing businesses [20]. Similarly, Cloutier proposed a reference architecture primer [64]. High-level guidelines for reference architecture design are also provided by Pohl et al. [65], whereas templates for reference architectures are often used in industrial organizations [66]. Angelov et al. developed a classification for reference architectures which could act as a starting point for defining reference architectures [26,67]. However, this "framework for reference architectures" focuses on types of reference architectures, rather than giving guidelines on how to construct a reference architecture. A more specific process for developing aspect-oriented reference architectures has been proposed in [68]. Taking previous work and the identified constraints into consideration, we define a generic framework for designing variability-intensive reference architectures, illustrated in Fig. 6. Bold boxes are part of a process proposed in [68] whereas regular boxes are additional steps and dotted boxes represent new steps performed as part of other steps. Arrows indicate flow, while constraints related to the steps of the process are shown in brackets.

As can be seen in Fig. 6, we explicitly consider quality attributes for variability-intensive service-oriented reference architectures, as identified in our study (variability, interoperability, scalability, etc.). Moreover, we include an explicit step to evaluate a reference architecture [69]. Most reference architectures reported in literature are not evaluated [6]. The evaluation includes the identification of mismatches between process and software. This framework is an example for how constraints can be used and how they affect reference architecture design. Developing a framework for reference architecture design is part of our ongoing work [25].

The identification of information sources is commonly done when designing reference architectures, even though it is often done implicitly rather than in a dedicated step. Legacy systems and external parties have to be identified in order to cope with the challenge of system integration and interaction with external parties. The use of open standards is considered as an architectural requirement. The identification of variability points is made explicit as part of establishing architectural requirements. Throughout the design of reference architectures, organizational thinking has to be ensured so that the organization adapts along with the introduction of the architecture. Similarly, the prevention of vendor lock-in needs to be considered throughout the design and introduction of a reference architecture (C8 is indirectly addressed through open standards and the use of open source).

6.3. Applicability of results in e-government

The results of our case study are not only applicable for e-government in the Netherlands, but can be applied to e-government in general. In the following we list some examples indicating that

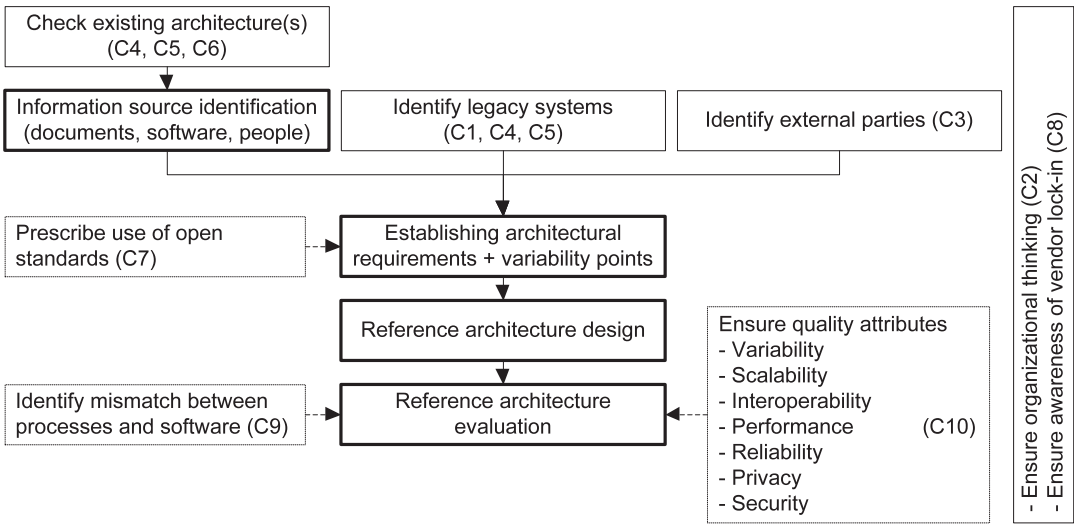

Fig. 6. Framework for reference architecture design for variability-intensive service-based systems.

SOA and reference architectures are becoming more and more popular in e-government not only in the Netherlands, but in many countries.

- There is a strong drive all over Europe towards service-orientation in the context of governments. For example, one of our industrial partners conducts similar e-government projects in Germany as well as in the Netherlands.
- International ICT standards for public administrations (like NORA in the Netherlands) exist in many countries. The SAGA standard (Standards and Architectures for e-Government Applications)[2] in Germany or TTSC in Great Britain have been developed. As the web provides new ways for citizens to interact with government agencies, e-government research has received considerable attention [70].
- Cross-national standards, such as the "European Interoperability Framework for Pan-European eGovernment" services[3] have been proposed.
- Throughout the world, in the public sector, service-oriented infrastructures have been proposed. Examples include the "Net-Centric Enterprise Services" from the United States Department of Defence (NCES),[4] the Austrian PVP ("Portalverbund Protokoll")[5], or the "German Administration Service Directory" (DVDV)[6] with an electronic service registry. Recently, a service-centric framework for digital government application has been proposed [71].

7. Validity

We classify limitations of this case study into construct validity, external validity and reliability [51,72]. Since the case study is exploratory and our study does not make any claims about causal relationships, internal validity is not a concern [51]. However, typical measures to address internal validity are discussed below in the context of construct and external validity as well as reliability.

Construct validity: Construct validity is concerned with whether or not we measured what is intended. We gathered data only from a limited number of sources and from one province in the Netherlands. However, our industrial partners also reported their experience with projects across the Netherlands and Europe. Moreover, the consistency of our data from three different data collection techniques gives us confidence that we identified real constraints that municipalities and vendors experience. Most constraints occurred in two or more data sources (see Table 3). Furthermore, the results from the interviews were checked by the interviewees to establish a chain of evidence.

External validity: External validity is concerned with the generalization of results and to what extent the findings are of interest outside the investigated case. Rather than aiming for statistical generalization, we aimed for analytical generalization. We were interested in constraints, rather than in generalization in terms of prediction or statistics as in social sciences [73]. We suggest that our case study provides sufficient details so that the generalization problem is a problem that can be addressed by identifying points of similarity between other projects and the characteristics of our case studied in this paper. Thus, to decide about the practical relevance and generalization of the constraints, the question is how the details reported in this paper are sufficiently similar to those in other projects [73]. Also, we conducted a single-case rather than a multiple-case design. However, as argued in Section 4, e-government is a representative case for variability-intensive environments: As we have found, e-government has similar stakeholders as other variability-intensive head-office organizations (i.e., organizations with a central office and many branches), such as banks or multi-national organizations. Furthermore, e-government shares characteristics with other head-office organizations that need to balance standardization and local variability. In multi-national organizations such as banks, branches in different countries serve the same purpose but are subject to different cultures, regulations, laws, staff structure, etc. To further mitigate the lack of generalizability, we identified the topic "Regulated industry" in Section 5.3 to show that some constraints are very specific to our case.

Moreover, when coding our data, we had to integrate data where different subjects might have used terms and concepts with different meanings [58]. Providing transcripts for comments to participants helped mitigate this problem.

Reliability: This aspect is concerned with how data analysis depends on the researchers. To address this issue, we prepared a case study protocol. Furthermore, results were reviewed by case subjects and more than one data collection technique and data

[2] http://www.cio.bund.de/cln_155/DE/Standards/SAGA/saga_node.html.
[3] http://ec.europa.eu/idabc/en/document/2319.
[4] http://www.disa.mil/nces/.
[5] http://portal.bmi.gv.at/ref/downloads/PVWhitepaper.pdf.
[6] http://www.bit.bund.de.

source was used. More than one researcher analyzed the data. Also, we found that all data sources provided us with more or less the same description of constraints. This could be an indication of data saturation. The case study involved people with different backgrounds, each expressing their experience and views, providing us with a coherent perspective on the topic at hand. Nevertheless, we found that the results obtained from the interviews, expert meeting and document analysis were consistent, which increases our confidence in the trustworthiness of the data.

Moreover, we have to critically appraise our own role in conducting the case study. We conducted the interviews and therefore might have influenced the answers (e.g., by asking questions that were of interest for us rather than focusing on topics that interviewees were most knowledgeable of). Also, we participated in the expert meeting and contributed to discussions. This could have caused biases in other participants of the expert meeting in terms of the topics and issues that were discussed.

Finally, the discussions on new variability-intensive reference architectures (Section 6.2.1), as well as the framework for designing variability-intensive reference architectures (Section 6.2.2) are based on the interpretation of the case study results outlined in Section 5. However, neither the notion of a new reference architecture nor the template for designing reference architectures have been validated independently on their own but are examples of how the results of the case study can be used.

8. Conclusions

We identified constraints for variability-intensive service-oriented reference architectures in regulated environments by conducting a case study in the e-government domain. We found ten concrete constraints and related these constraints to customers (municipalities) and software vendors. Furthermore, we identified which constraints are specific to the case study. We also discussed some current trends and future work in the context of service-oriented reference architectures.

The findings of our research provide insights that an organization (e.g., head-office organizations, such as banks, multi-national organizations) can use to design and implement variability-intensive service-oriented reference architectures. Our future research efforts are: The improvement of processes for the design of reference architectures, and the design of variability-intensive service-oriented reference architectures for e-government.

Acknowledgments

The authors thank all individuals and organizations that participated in the study. We also thank the anonymous reviewers for their valuable feedback. This research has been partially sponsored by NWO SaS-LeG, contract no. 638.000.000.07N07.

References

[1] M.H. Dodani, SOA 2006: state of the art, Journal of Object Technology 5 (2006) 41–48.
[2] M. Sinnema, S. Deelstra, P. Hoekstra, The COVAMOF derivation process, in: International Conference on Software Reuse, Springer Verlag, Turin, Italy, 2006, pp. 101–114.
[3] C. Sun, R. Rossing, M. Sinnema, P. Bulanov, M. Aiello, Modeling and managing the variability of web-service-based systems, Journal of Systems and Software 83 (2010) 502–516.
[4] M. Aiello, P. Bulanov, H. Groefsema, Requirements and tools for variability management, in: 4th IEEE Workshop on Requirement Engineering for Services (REFS 2010), IEEE Computer Society, Seoul, South Korea, 2010, pp. 245–250.
[5] T. Erl, SOA Design Patterns, Prentice Hall, Upper Saddle River, NJ, 2009.
[6] L.B.R. de Oliveira, K.R. Felizardo, D. Feitosa, E.Y. Nakagawa, Reference models and reference architectures based on service-oriented architecture: a systematic review, in: 4th European Conference on Software Architecture, Springer Verlag, Copenhagen, Denmark, 2010, pp. 360–367.
[7] R. Kazhamiakin, S. Benbernou, L. Baresi, P. Plebani, M. Uhlig, O. Barais, in: M.P. Papazoglou, K. Pohl, M. Parkin, A. Metzger (Eds.), Adaptation of Service-Based Systems, Springer Verlag, Berlin/Heidelberg, 2010, pp. 117–156.
[8] M. Galster, E. Bucherer, A business-goal-service-capability graph for the alignment of requirements and services, in: IEEE Congress on Services, IEEE Computer Society, Honolulu, HI, 2008, pp. 399–406.
[9] R. Haesen, M. Snoeck, W. Lemahieu, S. Poelmans, On the definition of service granularity and its architectural impact, in: 20th International Conference on Advanced Information Systems Engineering (CAiSE'08), Springer, Montpellier, France, 2008, pp. 375–389.
[10] Hadaytullah, K. Koskimies, T. Systa, Using model customization for variability management in service composition, in: IEEE International Conference on Web Services, IEEE Computer Society, Los Angeles, CA, 2009, pp. 687–694.
[11] F.M. Medeiros, E.S. de Almeida, S.R. de Lemos Meira, Towards an approach for service-oriented product line architectures, in: Workshop on Service-oriented Architectures and Software Product Lines, Software Engineering Institute, San Francisco, CA, 2009, pp. 1–7.
[12] Q. Gu, P. Lago, On service-oriented architectural concerns and viewpoints, in: Working IEEE/IFIP Conference on Software Architecture (WICSA), IEEE Computer Society, Cambridge, UK, 2009, pp. 289–292.
[13] Q. Gu, P. Lago, Exploring service-oriented system engineering challenges: a systematic literature review, Service Oriented Computing and Applications 3 (2009) 171–188.
[14] W. Anderson, What COTS and software reuse teach us about SOA, in: 6th International IEEE Conference on Commercial-off-the-Shelf (COTS)-Based Software Systems, IEEE Computer Society, Banff, AB, 2007, pp. 141–149.
[15] T. Boehmann, M. Junginger, H. Krcmar, Modular service architectures: a concept and method for engineering it services, in: 36th Annual Hawaii International Conference on System Sciences, IEEE Computer Society, Hawaii, USA, 2003, pp. 74b–84b.
[16] J. Lee, D. Muthig, M. Naab, An approach for developing service oriented product lines, in: 12th International Software Product Line Conference, IEEE Computer Society, Limerick, Ireland, 2008, pp. 275–284.
[17] OASIS, Reference Model for Service Oriented Architecture 1.0, 2006.
[18] OASIS, Reference Architecture for Service Oriented Architecture 1.0, 2008.
[19] R. High, S. Kinder, S. Graham, IBM's SOA Foundation – An Architectural Introduction and Overview, IBM, 2005, pp. 68.
[20] R. Cloutier, G. Muller, D. Verma, R. Nilchiani, E. Hole, M. Bone, The concept of reference architectures, Systems Engineering 13 (2010) 14–27.
[21] S. Angelov, P. Grefen, An E-contracting reference architecture, Journal of Systems and Software 81 (2008) 1816–1844.
[22] T.E. Faegri, S. Hallsteinsen, A software product line reference architecture for security, in: T. Kakola, J.C. Duenas (Eds.), Software Product Lines, Springer Verlag, Berlin/Heidelberg, 2006, pp. 275–326.
[23] M. Hafner, M. Memon, R. Breu, SeAAS – a reference architecture for security services in SOA, Journal of Universal Computer Science 15 (2009) 2916–2936.
[24] A. Grosskurth, M. Godfrey, A reference architecture for web browsers, in: International Conference on Software Maintenance, IEEE Computer Society, Budapest, Hungary, 2005, pp. 661–664.
[25] M. Galster, P. Avgeriou, Empirically-grounded reference architectures: a proposal, in: 7th International ACM Sigsoft Conference on the Quality of Software Architectures (QoSA), ACM, in press, Boulder, CO, 2011.
[26] S. Angelov, P. Grefen, D. Greefhorst, A classification of software reference architectures: analyzing their success and effectiveness, in: Joint Working IEEE/IFIP Conference on Software Architecture & European Conference on Software Architecture (WICSA/ECSA), IEEE Computer Society, Cambridge, UK, 2009, pp. 141–150.
[27] E.Y. Nakagawa, P.O. Antonino, M. Becker, Reference architecture and product line architecture: a subtle but critical difference, in: 5th European Conference on Software Architecture, Springer Verlag, Essen, Germany, 2011, pp. 207–211.
[28] O. Vogel, I. Arnold, A. Chughtai, E. Ihler, T. Kehrer, U. Mehlig, U. Zdun, Software-Architektur – Grundlagen – Konzepte – Praxis, Spektrum Akademischer Verlag, Berlin/Heidelberg, 2009.
[29] P. Clements, L. Northrop, Software Product Lines – Practices and Patterns, Addison-Wesley, Boston, MA, 2001.
[30] R. Hilliard, On representing variation, in: Workshop on Variability in Software Product Line Architectures, ACM, Copenhagen, Denmark, 2010, pp. 312–315.
[31] M. Galster, P. Avgeriou, Handling Variability in software architecture: problems and implications, in: 9th IEEE/IFIP Working Conference on Software Architecture, IEEE Computer Society, Boulder, CO., 2011, pp. 171–180.
[32] V. Stricker, K. Lauenroth, P. Corte, F. Gittler, S.D. Panfilis, K. Pohl, Creating a reference architecture for service-based systems – a pattern-based approach, in: G. Tselentis, A. Galis, A. Gavras, S. Krco, V. Lotz, E. Simperl, B. Stiller, T. Zahariadis (Eds.), Towards the Future Internet, IOS Press, Amsterdam, 2010, pp. 149–160.
[33] H. Choi, C. Lim, J. Kim, Defining reference architecture for NTIS development, in: 11th International Conference on Advanced Communication Technology, IEEE Computer Society, Gangwon-Do, South Korea, 2009, pp. 284–287.
[34] I. Futo, A functional IT reference model for public institutions, in: 29th International Conference on Information Technology Interfaces, IEEE Computer Society, Cavtat, Croatia, 2007, pp. 419–424.
[35] J. Leppaniemi, P. Linna, J. Soini, H. Jaakkola, Toward a flexible service-oriented reference architecture for situational awareness systems in distributed disaster knowledge management, in: Portland International Conference on

Management of Engineering and Technology (PICMET), IEEE Computer Society, Portland, OR, 2009, pp. 959–965.
[36] V. Peristeras, M. Fradinho, D. Lee, W. Prinz, R. Ruland, K. Iqbal, S. Decker, CERA: a collaborative environment reference architecture for interoperable CWE systems, Service Oriented Computing and Applications 3 (2009) 3–23.
[37] S. Reiff-Marganiec, H.-L. Truong, G. Casella, C. Dorn, S. Dustdar, S. Moretzky, The in context pervasive collaboration services architecture, in: Service Wave, Springer Verlag, Madrid, Spain, 2008, pp. 134–146.
[38] G. Costagliola, F. Ferrucci, V. Fuccella, Scorm run-time environment as a service, in: 6th International Conference on Web Engineering, ACM, Palo Alto, CA, 2006, pp. 103–110.
[39] Q. Zheng, B. Dong, F. Tian, W. Chen, A service-oriented approach to integration of e-learning information and resource management systems, in: 12th International Conference on CSCW in Design, IEEE Computer Society, Xian, China, 2008, pp. 1047–1052.
[40] J. Lee, G. Kotonya, Combining service-orientation with product line engineering, IEEE Software 27 (2010) 35–41.
[41] S. Cohen, R. Krut, Managing Variation in Services in a Software Product Line Context, CMU SEI, Pittsburgh, PA, 2010.
[42] M. Abu-Matar, H. Gomaa, M. Kim, A. Elkhodary, Feature modeling for service variability management in service-oriented architectures, in: 22nd International Conference on Software Engineering and Knowledge Engineering, KSI, Redwood City, CA, 2010, pp. 468–473.
[43] H. Gomaa, M. Saleh, Software product line engineering for web services and UML, in: 3rd ACS/IEEE International Conference on Computer Systems and Applications, IEEE Computer Society, Cairo, Egypt, 2005, pp. 110–113.
[44] S. Segura, D. Benavides, A. Ruiz-Cortes, P. Trinidad, A taxonomy of variability in web service flows, in: First Workshop on Service-oriented Architectures and Product Lines, SEI, Kyoto, Japan, 2007, pp. 1–5.
[45] A. Bouguettaya, A. Rezgui, B. Medjahed, M. Ouzzani, Internet computing support for digital government, in: M.P. Singh (Ed.), Practical Handbook of Internet Computing, CRC Press, Boca Raton, FL, 2004, pp. 1–14.
[46] J. Verner, J. Sampson, V. Tosic, N.A.A. Bakar, B. Kitchenham, Guidelines for industrially-based multiple case studies in software engineering, in: Third IEEE International Conference on Research Challenges in Information Science, IEEE Computer Society, Fes, Morocco, 2009, pp. 313–324.
[47] P. Vogt, Dictionary of Statistics and Methodology – A Non-technical Guide for the Social Sciences, Sage Publications, Thousand Oaks, CA, 2005.
[48] P. Runeson, M. Hoest, Guidelines for conducting and reporting case study research in software engineering, Empirical Software Engineering 14 (2009) 131–164.
[49] L. Bass, P. Clements, R. Kazman, Software Architecture in Practice, Addison-Wesley, Boston, MA, 2003.
[50] J. Gerring, Case Study Research – Principles and Practices, Cambridge University Press, Cambridge, NY, 2006.
[51] R.K. Yin, Case Study Research – Design and Methods, Sage Publications, London, UK, 2009.
[52] C. Robson, Real World Research: A Resource for Social Scientists and Practitioner-researchers, Blackwell Publishers, Oxford, UK, 2002.
[53] T.D. Bouma, Process Analysis and Requirement Specification of Software as Service for WMO Provision Applications at Dutch Municipalities, Faculty of Economics and Business, University of Groningen, Groningen, The Netherlands, 2010, pp. 131.
[54] C.B. Seaman, Qualitative methods in empirical studies of software engineering, IEEE Transactions on Software Engineering 25 (1999) 557–572.
[55] K. Krippendorff, Content Analysis: An Introduction to its Methodology, second ed., Sage Publications, Thousand Oaks, CA, 2003.
[56] A.C. Strauss, J. Corbin, Basics of Qualitative Research: Grounded Theory Procedures and Techniques, second ed., Sage Publications, Thousand Oaks, CA, 1990.
[57] M.B. Miles, A.M. Huberman, Qualitative Data Analysis, second ed., Sage Publications, Thousand Oaks, CA, 1994.
[58] G.W. Noblit, R.D. Hare, Meta-Ethnography: Synthesizing Qualitative Studies, Sage Publications, Newbury Park, CA, 1988.
[59] M. Galster, P. Avgeriou, Empirically-grounded reference architectures: a proposal, in: 7th ACM Sigsoft International Conference on the Quality of Software Architectures (QoSA), ACM, Boulder, CO., 2011, pp. 153–157.
[60] R. Weston, S. Kaviani, SaaS Vendor Selection – A Systematic Approach to Selecting a Software-as-a-Service Vendor, HyperOffice, 2009, pp. 12.
[61] E. Soederstroem, F. Meier, Combined SOA Maturity Model (CSOAMM): towards a guide for SOA adaption, in: R.J. Goncalves, J.P. Mueller, K. Mertins, M. Zelm (Eds.), Enterprise Interoperability II, Springer Verlag, Berlin/Heidelberg, 2007, pp. 389–400.
[62] R. Welke, R. Hirschheim, A. Schwarz, Service-oriented architecture maturity, IEEE Computer 44 (2011) 61–67.
[63] W.-J.v.d. Heuvel, O. Zimmermann, F. Leymann, P. Lago, I. Schiefendecker, U. Zdun, P. Avgeriou, Software service engineering: tenets and challenges, in: Workshop on Principles of Engineering Service Oriented Systems, IEEE Computer Society, Vancouver, BC, 2009, pp. 26–33.
[64] G. Muller, A Reference Architecture Primer, Embedded Systems Institute, Eindhoven, NL, 2010. pp. 1–21.
[65] K. Pohl, G. Boeckle, F. van der Linden, Software Product Line Engineering – Foundations, Principles, and Techniques, Springer Verlag, Berlin/Heidelberg, 2005.
[66] O. Winberg, Reference Architecture Template, Swedish Defence Materiel Administration, Stockholm, Sweden, 2007, pp. 23.
[67] S. Angelov, P. Grefen, D. Greefhorst, A framework for analysis and design of software reference architectures, Information and Software Technology 54 (2012) 417–431.
[68] E.Y. Nakagawa, R.M. Martins, K.R. Felizardo, J.C. Maldodano, Towards a process to design aspect-oriented reference architectures, in: XXXV Latin American Informatics Conference (CLEÍ2009), Pelotas, Brazil, 2009, pp. 1-10.
[69] S. Angelov, J. Trienekens, P. Grefen, Towards a method for the evaluation of reference architectures: experiences from a case, in: Second European Conference on Software Architecture, Springer, Paphos, Cyprus, 2008, pp. 225–240.
[70] J.L. Ambite, Y. Arens, W. Bourne, S. Feiner, L. Gravano, V. Hatzivassiloglou, E. Hovy, J. Klavans, A. Philpot, U. Ramachandran, K.A. Ross, J. Sandhaus, D. Sarioz, R.R. Schmidt, C. Shahabi, A. Singla, S. Temiyabutr, B. Whitman, K. Zaman, Data integration and access – the digital government research center's energy data collection (EDC) project, in: W.J. McIver, A.K. Elmagarmid (Eds.), Advances in Digital Government: Technology, Human Factors, and Policy, Kluwer Academic Publishers, Boston, MA, 2002, pp. 85–106.
[71] A. Bouguettaya, Q. Yu, Z. Liu, Z. Malik, Service-centric framework for a digital government application, IEEE Transactions on Services Computing 4 (2011) 3–16.
[72] C. Wohlin, M. Hoest, K. Henningsson, Empirical Research Methods in Software Engineering, in: R. Conradi, A.I. Wang (Eds.), Empirical Methods and Studies in Software Engineering, Springer Verlag, Berlin/Heidelberg, 2003, pp. 7–23.
[73] J. Hutchinson, M. Rouncefield, J. Whittle, Model-driven engineering practices in industry, in: 33rd International Conference on Software Engineering, ACM, Honolulu, HI, 2011, pp. 633–642.