SOA decision making - what do we need to know

Qing Gu, Hans van Vliet
Department of Computer Science
Vrije Universiteit Amsterdam, The Netherlands
{qinggu, hans}@cs.vu.nl

Abstract

The SOA decision making process addresses specific concerns, which requires (its own) specific information. Currently, SOA design decisions are often documented (if at all) using generic architectural design decision models. The information that is specifically relevant to SOA design decisions remains hidden in the description of the existing entities of architectural decision models. As a result, this information could be overlooked during the decision making process and result in unsound decisions. In this paper, we highlight a set of information items that aids SOA decision making.

1. Introduction

The importance of architectural decisions that emerge during the software architecting process has been commonly recognized [4, 5, 1]. These decisions can be regarded as knowledge assets that can be shared, discovered and reused in different software development projects. They should be captured and systematically documented to e.g. avoid knowledge vaporization [8].

Service-Oriented Architecture (SOA) is an architectural style that can be applied to construct software systems where heterogeneous services from a range of service providers are integrated [7]. The focus of engineering service-oriented systems therefore shifts from developing software applications to developing pools of services that can potentially be aggregated to provide a variety of solutions for various runtime scenarios to fulfill ever-changing business requirements.

As a result, specific architectural concerns arise in the design of service-oriented systems. For instance, when client-server is chosen as the architectural style for a software system, quality attributes like performance or maintainability are typically the primary architectural concerns; in the context of SOA, the scope of service pools, the runtime behavior of services, the cooperation with external stakeholders and governance issues become major concerns that need to be dealt with.

Similarly, specific information becomes relevant to the SOA design decision making process. Typically, the reasoning for selecting one of the design alternatives to form an architectural decision is based on (non-)functional requirements, business goals and cross-dependencies [3]. When making SOA design decisions, specific information (like the knowledge of external service consumers and the ownership of a business resource) is required. For instance, suppose an architect needs to decide between creating a new service and modifying the existing service to accommodate the new business requirements. The architect has to evaluate each alternative based on e.g. the impact on the existing service consumers and the potential cross-organizational cooperation. Therefore, the knowledge of external service consumers and the ownership of a business resource become particularly relevant.

In order to identify which information is relevant, we analyze the aspects that cause SOA concerns in the design of service-oriented systems. Based on the study of the literature, the knowledge of the state-of-the-art, and experience from various industrial case studies, we identified four aspects that are specific to SOA, namely a) temporary provision-consumption relationship, b) different architecture types, c) dealing with heterogeneity and d) different perspectives of stakeholders. From the analysis of these four aspects, we elicited a set of information items that are intensively involved in the SOA decision making process. From this set of information items, we identified eight knowledge entities that in our opinion should be explicitly expressed when documenting SOA design decisions.

To show the relevance of this information in SOA design decisions, we created two scenarios as tangible examples. In these scenarios, we discuss related SOA design decisions where the identified knowledge entities play important roles in the decision making process. In addition, we also study existing SOA patterns to get some evidence from proven SOA design solutions. From the problem space of the SOA design patterns, we observe that the four aspects are drivers.
for many SOA design issues. We also observe that the identified information is embedded in the description of the patterns. From these observations, we corroborate the relevance of the identified information.

In summary, the SOA decision making process addresses specific concerns, where specific information becomes relevant. The main contribution of this paper is to highlight this relevant information so that its modeling becomes possible. The modeling of this information benefits from the identification of SOA design decisions, the justification of alternatives and the classification of SOA design decisions. Accordingly, we argue that the representation of architectural decisions should be extended to leverage information relevant to SOA design decisions.

The remainder of the paper is organized as follows. The next section discusses the four aspects that may give rise to SOA design issues and information relevant to SOA design decisions. Section 3 describes two SOA design decisions based on two business scenarios to show the relevance of the elicited information and the benefits of being modeled as first class entities. In Section 4 we discuss our observations of published SOA design patterns and decisions in relation to the identified aspects and information. Related work on SOA design decisions is discussed in Section 5. We draw our conclusions and present an outlook for future research in Section 6.

2. Aspects relevant to SOA design

The design of service-oriented systems has to address specific design issues arising from the aspects that assume special relevance to service engineering. Some of these aspects are specific to service engineering, while some of them are less specific in the sense that they do share certain similarities with the traditional approaches (e.g. component-based development). All these aspects, however, are prominent and should be specifically considered in SOA design. In this section, we discuss four such aspects, where the first one is specific to service engineering and the rest three aspects are less specific.

2.1 Temporary provision-consumption relationship

In traditional software engineering, software applications are typically developed by software producers and installed on their customers’ computers. The execution environment, such as operating systems or databases, is known in advance (i.e. given in the installation requirements) by the customers. Further, the business relationship between the producers and their customers is usually long term and stable. As a result, the producers know how to inform their customers if new releases are available; and the end-users know how to contact the producer if faults are detected or requirements change.

In SOA, the way in which software applications (and services) are delivered is different in that service consumers use services but do not own these services. A temporary provision-consumption relationship exists between a service provider and its consumer. On the one hand, temporary refers to short-term in that the relationship exists only when a service consumer requests the service and terminates as soon as consumption is finished; on the other hand, temporarily implies dynamism in that a service may be consumed by different service consumers in different business solutions. Due to this temporary provision-consumption relationship, new SOA concerns arise with regard to runtime governance and uncertainty of service consumers.

- **Runtime governance** Since services physically reside at the service providers’ premises rather than the service consumers’, the service providers have to ensure that their services are available for consumption and perform as expected. To achieve this, the service providers have to monitor the behavior of the published services and manage their execution environment, which is not the case for traditional software applications. For instance, consider a situation in which the server where a software application or service is running encounters a system failure and requires a reboot. In the case of the software application, the user (customer) of the application has to take action to reboot the server; while in the case of service, the service provider has to reboot the server and make sure that the service is available again. Hence, an architect should be aware of this responsibility from the perspective of a service provider in the SOA decision making process.

- **Uncertainty of service consumers** Service providers may deliver services to different groups of consumers with different expectations under different runtime scenarios. The consumers and their expectations are not always known by the service provider at design time. Services are often designed according to some assumptions on service consumers (e.g. the number of concurrent invocations). As different assumptions might lead to completely different design decisions, these assumptions should be made explicit in the SOA decision making process.

Due to ever-changing business requirements, changes made to a service are inevitable. It is very un-desirable to interrupt an existing consumption or leave the improved functionalities unannounced. How to inform the existing service consumers about the changes becomes a design issue. Accordingly, assumptions on service consumers again become of great importance.
For instance, a service that assumes known service consumers may send a notification to its consumers whenever a change is made; while a service that assumes unknown service consumers may have to e.g. implement multiple service contracts to avoid interrupting existing consumers.

Hence, making assumptions on service consumers is necessary in the design of service-oriented systems. Accordingly, these assumptions should be explicitly considered in the SOA decision making process.

### 2.2 Different architecture types

The focus of engineering service-oriented systems is to create pools of services and to (re-)aggregate these services in such a way that different business requirements can be fulfilled. These services are leveraged as enterprise resources which can be shared across enterprises or even business partners. Each service in a service pool exposes a number of functionalities. Functionality delivered by various services can be aggregated according to business logic or rules to form composite services. Both services and composite services can be grouped to form one or multiple service pools depending on their business domain, ownership or governance responsibilities.

Hence, instead of designing one architecture for one software application, the design of service-oriented systems has to design architectures for services, composite services and services pools. Each type of these architectures faces different concerns.

- **Designing services** A service can be viewed as an independent software program that realizes a set of functionalities. As each service might have an architecture different from the others, it needs to be designed individually. The design of services includes the decisions on the functionalities it will deliver, its underlying infrastructures that support those functionalities and the design of a service contract.

  In order to decide the level of granularity and define the scope of services, architects (together with business analysts) have to deeply understand the *business process and its ownership* and available *IT support*. For instance, to ensure the integrity of a business solution, it would be desirable to embed the related functionalities in one service. However, if one of these functionalities participates in multiple solutions, a decision has to be made between isolating this functionality to create a separate service or leaving it as a repeated functionality among multiple services. The choice is heavily dependent on the relationship between solution logic, the ownership of business processes and the cost of creating new services.

The design of the underlying infrastructure of each service requires decisions on the physical environment within which it is deployed. These decisions require information about *enterprise resources and their ownership*, such as what resources the service needs to access and what resources are accessed by the other parts of enterprises as well.

Since services are invoked through their published contracts, service contracts become part of the design of services. From the perspective of a service provider, the design of a service contract requires assumptions on the service consumers (e.g. interest of service consumers). For instance, if the service is intended to be used by different groups of service consumers which have different requirements, multiple contracts designed specifically to each group of service consumers are preferred.

In summary, specific information relevant to designing services includes: the concerns from the perspective of a service provider, assumptions on the service consumers, information on business process and its ownership, *IT support* and enterprise resources and their ownership.

- **Designing composite services** The ultimate goal of designing services is to aggregate them to solve business problems. Composite services are the result of the aggregation, which are typically joined functionalities from participating services glued together with solution logic. The design of the corresponding architecture of composite services has a different focus than e.g. a component-based architecture in that service participants are distributed with different owners, and various composition configurations should be designed to accommodate various runtime scenarios.

Since service participants representing units of business process are often distributed within the enterprise owned by different stakeholders, the composition designer often needs to collaborate with these stakeholders to e.g. pose additional requirements. Therefore, being aware of *business process and its ownership* is sometimes necessary in the design of composite services.

Further, it should be noticed that a service composition can be both a service provider and service consumer depending on its runtime behavior. It is a service provider when it publishes its service contract for consumption; and it becomes a service consumer when it invokes its service participants. As discussed in Section 2.4, these two roles have different concerns. Hence, a composite service should be designed from the perspective of both a service consumer and provider.
In summary, being aware of the concerns from the two perspectives and the information of business process and its ownership are relevant to SOA decision making.

- **Designing service pools** Ideally, an enterprise maintains one service pool that comprises resources needed for a complete list of business requirements. However, in reality (especially in large enterprises having many departments), it is difficult to capture all the business processes and rules to create business models that cover the entire enterprise. The lack of complete and concrete knowledge about these business models hinders the success of collecting a complete list of business requirements. Hence, a service pool for an entire enterprise is often hard to implement.

  Instead of creating one service pool, another option is to build multiple service pools, each of which is governed by a specific enterprise department unit. The division of service pools is often driven by enterprise resources and their ownership (e.g. business domains) and IT support (e.g. various IT groups supporting various business functions). In this way, an enterprise may gradually adopt SOA depending on its goals and capabilities by allowing more flexibility for the sake of convenience or historical influences.

Although allowing multiple service pools with variations (in terms of implementation technologies, standardizations and transmission protocols) may increase the efficiency of building each pool, problems may occur when a service in one pool tries to invoke services in other pools. Therefore, the boundaries of service pools need to be well designed such that the chance of cross-boundary invocation is minimized.

Hence, the decision to place services (and service compositions) in a certain service pool depends on multiple concerns in relation to enterprise resources and their ownership and IT support, which assume special importance in service engineering.

### 2.3 Dealing with heterogeneity

SOA by definition allows heterogeneity in service-oriented systems. Often, services are designed independently in different projects, by different authorities and using different technologies. Eventually, when these services are required to be aggregated to fulfill business requirements, inconsistent design, implementation, data format, protocol, service contract, business rules and policies may bring lots of transformation effort. Therefore, how to deal with the unavoidable heterogeneity and how to avoid potential heterogeneity in the design are main concerns.

Since the heterogeneity often occurs when aggregating services from different pools and different owners, it is extremely important to know which service pool each of the participating services resides. Hence service pools as an architecture type assumes relevance to SOA decision making.

#### 2.4 Different perspectives of stakeholders

In SOA, services are collaboratively designed and developed by multiple SOA roles, such as service consumer and service provider. These roles often face different concerns.

From the perspective of a service provider, the designer may be concerned with the deployment environment of the services, the privacy and security related issues, the service contract to be published and expected service consumers. While from the perspective of a service consumer, the designer may be concerned with the development effort, the selection of services and with potential collaboration between the owner of the services.

Often a SOA design decision involves multiple (composite) services and accordingly multiple service providers and consumers may coexist in relation to these (composite) services. Without being aware of the roles, important concerns could be overlooked. Therefore, being aware of the different concerns from the two perspectives shows relevance to SOA decision making.

#### 2.5 A summary of identified information

Up to now, we have discussed four service aspects and elicited a set of information items relevant to the SOA decision making process in relation to each aspect. From this information, we identified the eight knowledge entities (listed in Table 1) relevant to the SOA decision making process. The exemplified application of these entities can be found in Section 3.

<table>
<thead>
<tr>
<th>Table 1. Identified knowledge entities</th>
</tr>
</thead>
<tbody>
<tr>
<td>1 The type of the architecture</td>
</tr>
<tr>
<td>2 Enterprise resources and their ownership</td>
</tr>
<tr>
<td>3 Business process and its ownership</td>
</tr>
<tr>
<td>4 IT support</td>
</tr>
</tbody>
</table>

#### 3 Two examples of SOA design issues

Due to the distinctive aspects that have been discussed in Section 2, business constraints and quality attributes are not
the only factors that are used in the rationale for SOA design decisions. Instead, the scope of services, the knowledge of consumers and the level of heterogeneity all influence SOA design decisions. In the following, we present two examples to show how these elements aid SOA decision making.

3.1 Example 1

Consider a scenario where the financial department of a retailer has to create a monthly “Profit & Loss statement” by determining the cost (e.g. cost of sales, marketing and sales expenses and general and administrative expenses) and the income (e.g. revenue from sold products and sales commissions). The IT department has created two services: a Cost service and Income service. These two services are aggregated to form a composite service called P&L statement which specifies the profit or loss realized in a period. The head-office of this retailer has noticed that each month there is a huge gap between the budget (delivered yearly) and the profit reported each month. Therefore, it requests the retailer to add a variance analysis to the monthly P&L statement so that the actual results can be monitored against budget.

![Diagram showing the composition of services](image)

Figure 1. Example 1: a) change the composite service, b) add a new service composition

The architect in the IT department faces two alternatives: a) change the Profit service by adding a newly created Budget service; or b) keep the Profit service and use it as a sub-composition logic for a new composite service: P&L variance analysis (actual versus budget). These two alternatives are illustrated in (Fig. 1.a) and (Fig. 1.b) respectively. The straight lines in the Figure represent the existing situation, while the dashed lines indicate the changed or new situation. Comparing these two alternatives, we observe that 1) the first alternative potentially delivers higher performance than the second one, but the service contract requires change; 2) the second alternative increases the reusability of the P&L statement service but decreases performance and brings more governance effort since an additional layer of service composition is introduced.

Critical to decision making is the knowledge of the existing service consumers. If the existing service consumers are unknown or outside the boundary of the financial service pools (e.g. other departments or retailers), the second alternative is preferable because it does not influence current agreements with existing consumers. If the service consumers of the P&L statement service are within the boundary of the financial service pool, the service architect might prefer the first alternative. However, adding budget information to the P&L statement may not be desired by all service consumers. The modified representation of the statement could lead to unexpected results, e.g. a bonus service that calculates the bonus for employees based on the output of the P&L statement may encounter failure if the structure of the P&L statement is completely changed. Further, the architect has to consider the feasibility of informing all the service consumers about the change.

In this example, besides the quality attributes (like governance effort, reusability, performance) that have been used in making the trade-off analysis, the knowledge about current agreements with consumers becomes the most important factor in choosing for one of the alternatives. Therefore, whether the service consumers are known or unknown and whether they are internal or external should be explicitly documented.

3.2 Example 2

Suppose now a new business requirement requests the P&L variance analysis to explicitly show the cost of marketing activities. As all the data related to marketing activities is maintained by the marketing department through the Marketing activities service, the P&L variance analysis service therefore faces the challenge of composing services distributed over two service pools. The architect hence faces two alternatives: a) let Marketing activities be exposed to external consumers and directly invoked by P&L variance analysis (Fig. 2.a); or b) create Published marketing activities as an intermediary service with a service contract specifically designed for the financial department and exposes only the functionalities that are needed by the financial department (Fig. 2.b).

The first alternative faces privacy and security-related concerns since the Marketing activities service might comprise functionalities that should not be exposed to external consumers due to e.g. confidentiality reasons. Further, external service consumers may not appreciate all the functionalities when they are only after a subset of the func-
Figure 2. Example 2: a) create an intermediary service, b) direct invocation

Figure 2. Example 2: a) create an intermediary service, b) direct invocation

From the service provider’s perspective, the second alternative provides better protection for the marketing department in terms of privacy and security since business sensitive functionalities can be shielded from external consumers. In addition, the possible changes made on service Marketing activities do not have impact on external consumers since the service Published marketing activities remains the same. However, by introducing a new service, the second alternative requires more development and governance effort. From the service consumer’s perspective, the P&L variance analysis service invokes the Published marketing activities service through a dedicated service contract designed specifically for the financial department. Without being overwhelmed by all the functionalities delivered by the Marketing activities services, the P&L variance analysis service may perform the invocation more efficiently.

In this example, besides the quality attributes (like privacy, development effort, security) that have been justified in the rationale for each alternative, the perspective of a service provider or service consumer, the ownership of business resources and the boundaries of service pools are all relevant. Therefore, we argue that this information should be explicitly documented in this SOA design decision.

3.3 Foreseen benefits

The two SOA design issues and related knowledge (illustrated in the previous Section) have been formalized in Fig. 3.a and Fig. 3.b. To this end we used the knowledge entities identified in Section 2.5.

From these two examples, we observe that by modeling them as first class entities, we can support the following objectives:

- **Facilitate the detection of SOA design issues.** Due to still emergent best practices and knowledge sharing in the context of SOA, the identification of SOA design issues is still a challenge [10]. Sometimes, design issues that are relevant can be overlooked due to e.g. the lack of experience in SOA design. It has been extensively studied and proven that making explicit the information about architectural design decisions facilitate the decision making process [1]. In the SOA context, this can further help in acquiring insight and accelerating maturity.

For instance, the three knowledge entities: enterprise resources and their ownership, business process and its ownership and IT support may be associated with different stakeholders each participating in SOA design decisions. If made explicit, potential collaboration between these stakeholders becomes more obvious. Consequently, this collaboration may highlight new related design issues.

- **Highlight the relevant information for justifying alternatives.** As shown in Fig. 3, the identified knowledge entities play important roles in justifying the selection of design alternatives. When information relevant to SOA decision making is explicitly expressed, this information will likely to participate in the evaluation of each alternative. In this way, the SOA design decisions are more sound and convincing.

- **Enable filtering specific design decisions based on certain criteria.** By explicitly documenting the relevant information for SOA design decisions, it becomes possible to select a group of SOA design decisions that have the same characteristics. These characteristics form a set of indirect relationships between SOA design decisions, which is complementary to the direct relationship that is usually documented in architectural decisions already. For instance, one can have a selection of SOA design decisions which are related to services designed for external service consumers or services that require access to certain enterprise resources. By doing so, a pattern language may be established.

4 SOA design decisions in the literature

Since the aspects we identified are extracted from our observations and the relevant information is deduced from the aspects, we need to verify whether this information is indeed used in SOA design decision making in practice. Therefore, we studied two types of SOA design decisions:
SOA design patterns and SOA design decisions published in projects.

Most of the SOA design patterns can be regarded as general SOA design decisions without a specific context. They become proven design solutions to common problems related to the aspects discussed in Section 2. For instance, among the 85 SOA design patterns categorized in [2], 17 patterns deal with the inconsistency between data, contract, protocol, policies, business rules, design and implementation (this corresponds to aspect dealing with heterogeneity presented in Sec. 2.3); 13 patterns deal with the definition of services, composite services and service pools (aspect: different architecture types in Sec. 2.2); 18 patterns deal with the uncertainty of service consumers (aspect: temporary provision-consumption in Sec. 2.1). We observe that most of the documented patterns are (implicitly) presented from the perspective of a service provider (aspect: Different perspectives of stakeholders in Sec. 2.4). This aspect crosscuts all the other aspects. In summary, more than 50% of the patterns in [2] address the four aspects we identified.

Another set of SOA design patterns described in [6] is extracted from a Web Services Composition Server project. All these patterns deal with how to deliver services to unknown service consumers, including the access to services, protecting services from untrusted consumers, decoupling known information from unknown information to improve response time, and so forth. These patterns correspond to the aspect temporary provision-consumption discussed in Sec. 2.1.

Furthermore, we also studied some SOA design decisions published in the literature [11, 9]. We notice that many of these design decisions focus on very low-level issues (e.g. mechanisms for dynamic composition and transactions). While these definitely belong to SOA standards and technologies, and to some extent may influence the resulting (technical) architecture, they miss the ‘big picture’ and therewith fail to participate in the more conceptual architectural aspects. Therefore, based on the material publically available it is very difficult to judge the extent to which they address aspects relevant to SOA.

5. Related work

The ways to capture, manage and model architectural design decisions have been well known research topics in the community of architectural knowledge management. Our work mainly focuses on design decisions in the context of SOA. Being domain-specific, SOA design decisions still share common use cases (e.g. discovery, sharing) with architectural knowledge management in general.

The information that facilitates SOA decision making is not new per se. Instead, this information appears in ar-

Figure 3. Documenting SOA design decisions with identified knowledge entities: a) change the composite service, b) add a new service composition

<table>
<thead>
<tr>
<th>Knowledge entities</th>
<th>Alternative A</th>
<th>Alternative B</th>
</tr>
</thead>
<tbody>
<tr>
<td>Related service</td>
<td>P&amp;L variance analysis</td>
<td></td>
</tr>
<tr>
<td>The type of the architecture</td>
<td>composite service</td>
<td></td>
</tr>
<tr>
<td>Enterprise resources and their ownership</td>
<td>Financial database owned by the financial dept.</td>
<td></td>
</tr>
<tr>
<td>Business process and its ownership</td>
<td>Financial report process owned by the financial dept.</td>
<td></td>
</tr>
<tr>
<td>IT support</td>
<td>IT group within the financial dept.</td>
<td></td>
</tr>
<tr>
<td>Types of service consumers</td>
<td>external</td>
<td></td>
</tr>
<tr>
<td>Assumptions on service consumers</td>
<td>This service is consumed by the head office, and possibly by unknown consumers from other departments.</td>
<td></td>
</tr>
<tr>
<td>The perspective of a service provider</td>
<td>Less effort and cost to implement the change, service contract changes, the service consumers may encounter unexpected results</td>
<td>More effort and cost since a new composite is created, increased instability</td>
</tr>
<tr>
<td>The perspective of a service consumer</td>
<td>Higher performance</td>
<td>Lower performance</td>
</tr>
<tr>
<td>Marketing activities</td>
<td>Marketing activities</td>
<td>Published marketing activities</td>
</tr>
<tr>
<td>The type of the architecture</td>
<td>Service</td>
<td>Service</td>
</tr>
<tr>
<td>Enterprise resources and their ownership</td>
<td>Marketing strategy owned by the marketing dept.</td>
<td>Marketing strategy owned by the marketing dept.</td>
</tr>
<tr>
<td>Business process and its ownership</td>
<td>Marketing plan owned by the marketing dept.</td>
<td>Financial report process owned by the financial dept.</td>
</tr>
<tr>
<td>IT support</td>
<td>IT group within the financial dept.</td>
<td>A new IT group that is responsible for all the intermediary services</td>
</tr>
<tr>
<td>Types of service consumers</td>
<td>Internal, known</td>
<td>External, unknown</td>
</tr>
<tr>
<td>Assumptions on service consumers</td>
<td>This service is consumed only by the marketing department</td>
<td>This service is consumed only by the financial department</td>
</tr>
<tr>
<td>The perspective of a service provider</td>
<td>Increased privacy and security-related issues</td>
<td>More effort and cost since a new service is created.</td>
</tr>
<tr>
<td>The perspective of a service consumer</td>
<td>N/A</td>
<td>Higher performance</td>
</tr>
</tbody>
</table>
chitectural decisions in other domains as well. The difference lies in the fact that this information actively participates in SOA decision making rather than being presented as pieces of architectural knowledge. For instance, being aware of different stakeholders with different concerns is well-known; and different viewpoints are typically used to describe the software architecture to different stakeholders to address their concerns. In SOA design decisions, stakeholders, which are associated to the owners of enterprise resource, business process and IT support, are regarded as valuable input to reason about a SOA decision (e.g. an alternative is rejected because sharing certain resources among certain stakeholders is against certain enterprise policies).

In [10], Zimmermann et al. present an approach to effectively capture design decisions and propose an architectural decision model to facilitate SOA decision making. As our work shares the similar goal but focuses on the relevant information that should be captured, it can be regarded as complementary to the work in [10]. Further, in the model proposed in [10], some of the elements (e.g. scope, roles) are comparable to some of the identified information (e.g. architecture types, perspectives) in this paper. However, scope and roles are used for administrative purposes rather than the evaluation of design alternatives.

6. Conclusion

The design of service-oriented systems focuses on the design of a set of services and their compositions, as well as placing them into service pools to fulfill ever-changing business requirements. Accordingly, specific design issues emerge due to some aspects that assume great importance to service engineering. We specifically discussed four aspects, namely: a) temporary provision-consumption relationship, b) different architecture types c) dealing with heterogeneity and d) different perspectives of stakeholders.

From the discussion of these aspects, we identified a set of information items that is relevant to SOA decision making. In addition, we presented two business examples with example SOA design decisions to show how this information provides useful input to the rationale of SOA design decisions.

The relevance of this information is not limited to its usefulness for reasoning about SOA design decisions. When this information is leveraged as first class entities in SOA design decision models, it can facilitate the identification of SOA design issues, increase the chance of making sound decisions, enable the identification of implicit relationship between decisions and provide more dimensions for the classification of SOA design decisions. Therefore, the relevance of identified information becomes clearer.

With the aim of gaining more insight into the identified information and its relevance to SOA decision making, in our future work we plan to conduct industrial case studies to capture more concrete SOA specific design decisions. By doing so, we are able to validate the information items identified in this work and to gain more experience of how these information items can be captured. Further, we intend to extend the existing architectural decision models to accommodate the identified information.

Acknowledgments

Parts of the research leading to these results have received funding from the FP7’ grant agreement 215483 (SCube European Network of Excellence in Software Services and Systems).

References