Towards the Future Internet
G. Tselentis et al. (Eds.)
IOS Press, 2010

doi:10.3233/978-1-60750-539-6-149

Creating a Reference Architecture for Service-Based Systems - A Pattern-Based Approach

Vanessa Stricker[a], Kim Lauenroth[a], Piero Corte[b], Frédéric Gittler[c], Stefano De Panfilis[b], and Klaus Pohl[a]
[a] *University of Duisburg-Essen, Germany*
[b] *Engineering Ingegneria Informatica S.p.A., Italy*
[c] *HP Hewlett Packard European Laboratories – Bristol, UK*

Abstract. The variety of technologies and standards in the domain of service-based systems makes it complex to build architectures which fit specific project contexts. A reference architecture accompanied by guidelines for deriving context-specific architectures for service-based systems can ease this problem. The NEXOF-RA project is defining a reference architecture for service-based systems that serves as a construction kit to derive architectures for a particular project context. Experience in developing the reference architecture over the last two years has shown that the service-oriented context results in different and sometimes contradicting demands for the reference architecture. Therefore, the development of a single and integrated reference architecture is not feasible. Instead, for constructing the reference architecture, the project has chosen a pattern-based approach that allows the consideration of different types and demands of service-based systems. Thus it can deal with contradicting demands of different types of service-based systems and is extensible to include new future trends of service-based systems. This paper will present the structure of the pattern-based reference architecture and explain how it addresses the needs of a reference architecture for service-based systems.

Keywords. Service-Based Systems, Service-Oriented Architecture, Reference Architecture, Pattern-Based Reference Architecture.

1. Introduction

Shaping the Future Internet around the new service-oriented paradigm has become an important task in research and industry, especially, considering the importance of this infrastructure as information, service, and networking means to the overall society world-wide. Taking the open world assumption of the Internet into account, the path towards the Future Internet reveals an omnipresent trend towards the integration and federation of heterogeneous service-based systems (SBS) from various domains. Different trends like software as a service, cloud computing, Internet of Services, Internet of Things, and web 2.0/3.0 result in the need for different types of architectures to support these different SBSs. Thus, the Future Internet needs to be architected in a way that allows the integration and federation of these SBSs.

Considering the large number of technologies and standards that emerged to address different aspects of architectures for SBSs, defining a specific architecture that

meets designated requirements is not an easy task. A reference architecture for SBSs as it is created within the NEXOF-RA [1] project under the umbrella of the European Technology Platform NESSI [2] (Networked European Software & Services Initiative) will allow providing guidance in this process. NEXOF, the NESSI Open Service Framework, is "[…] an integrated, consistent and coherent set of technologies and associated methods and tools […]".

The need to capture and address the different (sometimes contradicting) requirements of different types of SBSs and their characteristics has resulted in the adoption of a pattern-based approach within the NEXOF-RA project. The pattern-based approach has been implemented to allow the derivation of specific architectures for specific contexts in different domains following the generally known idea of architectural or design patterns (cf. [3], [4]). In this sense, the approach fosters the creation of a reference architecture as a system of patterns as described by Buschmann et al. (cf. [5], [6]) that can be composed to a specific architecture according to specific requirements. Accordingly, each different type of SBS is described by a system of patterns. In comparison to other existing reference architectures such as the OASIS reference architecture, this pattern-based reference architecture allows the integration and federation of different types of SBSs as well as the possibility to cope with new emerging trends in the future.

In its ambition to provide a reference architecture that allows the easy derivation and creation of specific architectures, the pattern-based reference architecture can be seen as a construction kit that provides a system architect with necessary instruments. This ambition raises different expectations going beyond the scope of traditional, well-known architectures. This paper presents the structure of the reference architecture that has been defined around the pattern-based approach considering all these goals.

In order to develop a reference architecture that addresses the creation of SBSs the first step is to choose a structure for this reference architecture. This means to understand the goals the reference architecture needs to satisfy and to identify the elements that constitute the reference architecture. Related work that addresses these questions for traditional reference architectures in software engineering is discussed in Section 2. Section 3 discusses the implications for the pattern-based reference architecture based on the related work and explains why its goals go beyond traditional reference architectures known in the software engineering community. In Section 4 the resulting structure of the pattern-based reference architecture, its individual elements and their relationships is explained. Furthermore, the relevance of the different elements for deriving specific architectures is explained. In Section 5 finally, the presented results are discussed critically and an outlook on future work is given.

2. Structures of Traditional Reference Architectures

Reference architectures and the definition of their content have a long tradition in software engineering as well as information science research. They play an important role because of several reasons. Different authors classify them in a slightly different way but one main reason for their usage that all definitions have in common, is the notion of reuse to ease the creation of new systems by building upon proven knowledge

[1] www.nexof-ra.eu
[2] www.nessi-europe.com

and experience. By proposing a framework for a systematic reuse of knowledge and software they can for example shorten development time while increasing the quality. "As shown for the reuse of software artifacts in software engineering, it is assumed that this reuse is especially efficient if it is not performed ad hoc but systematic, planned, and supported by sound methods." [11]

Reference architectures can be refined and adapted to derive different specific architectures [7]. Bass et al. [10] also consider reference models to be an important artifact accompanying reference architecture construction and usage: "A reference model is described as a division of functionality together with data flow between the pieces. […] Arising from experience, reference models are a characteristic of mature domains. [...] Whereas a reference model divides the functionality, a reference architecture is the mapping of that functionality onto a system decomposition."

In the following two different classifications of reference architectures and their characteristics are presented which served as a basis for deriving the goals, and the structure of the pattern-based reference architecture.

2.1. Beneken

Beneken [8] defines a reference architecture as an abstract software architecture that defines structures and types of software elements as well as their possible interactions and responsibilities applicable for all systems in this domain. Beneken distinguishes between three different types of reference architectures.

- **Functional reference architectures** separate the functional range of systems into logical functional concerns. The collaboration and the data flow between those concerns as well as their responsibilities and hierarchical dependencies are specified.
- **Logical reference architectures** define structures using layers and components as well as their hierarchy and communication dependencies. Defining the structure in an implementation manner without making implementation decisions or using specific technologies is the primary focus of this type.
- **Technical reference architectures** define components as a unit of implementation and deployment and determine a specific technology to realize them. Several technical reference architectures can adhere to one logical reference architecture. Components that are defined in the technical reference architecture refine components of the logical reference architecture.

These different types of reference architectures are designed to achieve different goals. While a functional reference architecture will be relevant in the early phases of the software development, a technical reference architecture will most likely be used when the detailed architecture needs to be specified and implemented. Beneken also distinguishes the following different elements and related views that should be documented by the reference architectures:

- Architectural overviews only provide an informal description of the coarse-grained structure of the software systems in a specific domain.
- Textures describe the structures, principles, and design concepts that are frequently used in software systems of a specific domain. Structures of components can be described in terms of their responsibilities, communication dependencies, and potential interfaces. Principles that address crosscutting aspects can be described, by defining design and implementation rules as well as policies.

- Reference interface specifications describing external visible behaviour allow components to become independent of any implementation and thus exchangeable.
- Infrastructures described by a reference architecture constraint the communication dependencies between components as well as the behaviour of the system during runtime. This view defines which basic services are provided by the used resources.

While functional reference architectures most likely only provide an architectural overview and name some of the interfaces, the logical and technical reference architectures provide views for the overview, the textures, the interfaces as well as the infrastructure focusing on different aspects. The infrastructure, for example, would only be described in its fundamental elements in a logical reference architecture while it would be fully and precisely described in a technical reference architecture.

2.2. Vogel et al.

Vogel et al. [9] also define reference architectures as a means to combine general architectural knowledge and experiences with specific requirements to derive an architectural solution in a certain context: "They [reference architectures] define structures of systems, essential building blocks, their responsibilities, and their collaboration." Reference architectures can be used by a system architect to derive a specific architecture specification. They differentiate between different types of reference architectures focusing on their usage context:

- platform-specific reference architectures should be adopted without any adaptation
- industry specific reference architectures focus on the needs of companies in a specific area
- industry crosscutting reference architectures cover more than one industry
- product line architectures describe architectures for similar software products

The description of the reference architectures should be based on well known and proven principles, types, and patterns. For each reference architecture a reference model should be defined that captures the functionalities and the information flow between functional blocks of a problem space addressed by a certain reference architecture. The reference architecture specifies how functional building blocks are distributed on system building blocks as well as their responsibilities and collaborations. In order to enable the usage of the reference architecture, sufficient documentation and guidelines towards a stepwise adaptation should be provided. These documentations should provide a mapping between the reference model and the reference architecture and describe which architectural means, decisions, and impacts have been considered.

3. Towards the Pattern-Based Reference Architecture

The basic research on reference architectures presented in the last section was used within the NEXOF-RA project to identify the implications for the pattern-based reference architecture for SBSs.

3.1. Implications for the Reference Architecture

On the one hand, the reference architecture for SBSs should provide a framework that allows the integration of research results, allowing the identification of gaps in the

current solution landscape. To construct such a reference architecture, existing solutions, standards and technologies should be used and combined in a way that reasonable system architectures can be derived. This is a crucial aspect of the reference architecture since there are already hundreds of standards and partial solutions out there and there is no need to invent from scratch.

On the other hand, the reference architecture aims at being a construction kit from which specific architectures can be derived allowing SBSs to be integrated, federated, and also to be implemented. The ambition of the pattern-based reference architecture is that all steps that need to be performed from the requirements to the real implementation of a SBS can be supported by providing all these architecture views. The pattern-based reference architecture furthermore tries to address several domains and not only one as it is generally assumed for reference architectures. Although, all of the addressed architectures are defined for SBSs, these systems can be very different in their goals, scope, and needs.

Considering these ambitions the reference architecture developed in the NEXOF-RA context cannot be classified as one of the above mentioned types (cf. Beneken [8]) of reference architectures without any overlaps. It needs to specify different views providing functional aspects, a logical decomposition but also technical information. The elements addressed by Beneken in the architectural overview, the textures, the interfaces, and the infrastructure views should all be provided in the pattern-based reference architecture.

In order to cope with these needs a well defined structure has been developed for the reference architecture. The pattern-based reference architecture is aimed to be domain independent and will be accompanied with a sound methodology and tools to be properly instantiated into a broad range of application domains by a number of end-user communities (including Large, Medium, and Small Enterprises) on different technologies. This reference architecture will mainly consist of a set of integrated specifications recommending different solutions.

The central part of the reference architecture depicts a system of patterns as described by Buschmann et al. (cf. [5], [6]) that allows the integration of different families of systems addressing different types of SBSs, e.g. a family for Enterprise SOA or the Internet of Services. However, the specification of this pattern system is not sufficient to provide a valuable reference architecture from which specific architectures can be derived. It is accompanied with several other elements that foster the instantiation.

3.2. The Pattern-Based Approach

One of the fundamental principles of software engineering that is largely used when designing a software system is known as the “separation of concerns”. In short, this principle states that a larger problem is more effectively solved when decomposed into a set of smaller problems or concerns. This gives software engineers the option of partitioning solution logic into capabilities, each designed to solve an individual concern. Related capabilities can be grouped into units of solution logic. The main benefit of solving problems this way is that a number of the solution logic units can be designed to solve immediate concerns while still remaining agnostic to the greater problem. This provides the constant opportunity to reuse the capabilities within those units to solve other problems as well.

The NEXOF-RA approach is based on the principle of separation of concerns. The objective of the pattern-based reference architecture is to address the problem of specifying service-based software system architectures by partitioning the overall solution into several pieces of the design solution: patterns. In particular, the need for a development methodology to develop large-scale complex systems and, at the same time, learn from the experiences of other system designers in solving recurring design problems has been recognized. "A pattern can be thought of as a set of constraints on an architecture – on the element types and their patterns of interactions – and these constraints define a set or family of architectures that satisfy them." [10]

In 1994 the Gang of Four already recognized the need to make a design evolvable and proposed the usage of patterns to ensure this: "The key to maximize reuse lies in anticipating new requirements and changes to existing requirements in designing your systems so that they can evolve accordingly. To design the system so that it's robust to such changes, you must consider how the system might need to change over its lifetime. […] Design patterns help you […] by ensuring that a system can change in specific ways. Each design pattern lets some aspect of system structure vary independently of other aspects, thereby making a system more robust to a particular kind of change." [4]

Usually, the documentation of design patterns, as it stands, describes details about the problem space that needs to be addressed as well as an according specific design/architectural solution for a sub-system, i.e. its structure, component behaviors, component interaction, and global properties. Furthermore, the solution provided by a pattern is given in terms of architectural choices and statements that claim how these architectural choices affect the quality attributes of a system that is compliant to such a pattern. Bass et al. [10] have stated that "One of the most useful aspects of patterns is that they exhibit known quality attributes. This is why the architect chooses a particular pattern and not one at random." Effects on a set of predefined quality attributes are clearly stated within the pattern descriptions in order to foster the derivation of adequate architecture instances.

The approach is also focused on how to compose these patterns together to develop complete systems. A complete system cannot nor will ever be built from a single pattern. It is the integration and composition of patterns that makes a whole system. Therefore, a structural approach to use patterns as first class design elements is adopted, i.e. they can be used in the design of a system as any other design element: class, module and component. This kind of patterns is called constructional patterns.

A constructional pattern is an architectural/design pattern with additional constraints that allow their composition and integration. A constructional pattern is a first-class design element that encapsulates a solution to a frequently recurring design problem, it hides lower level design decisions, and it offers interfaces to other design artifacts. In this sense, a constructional design pattern becomes a design component with interfaces. Specifying a pattern as a design element leverages the interest in a pattern to a higher design level that hides later design details and preserves consistency with lower levels. The structural approach includes three types of patterns pre-defining a hierarchy among different patterns.

1. Top-level patterns describe the characteristics of service framework families. Currently three different types of families are under construction to be integrated into the pattern-based reference architecture in the NEXOF-RA context: the Enterprise SOA (ESOA) top-level pattern, the Internet of Services top-level pattern and the Cloud Computing top-level pattern. Other system families addressing for

example the Internet of Things can easily be integrated into the reference architecture as a new system of patterns.

2. The top-level patterns are refined by abstract design patterns which refer to abstract components and patterns. They can be defined and refined on several levels of abstraction until at least one specific component becomes part of the solution.
3. Patterns referring to specific implementation components are called implementation design patterns. The ESOA system of patterns currently includes 42 patterns beside the ESOA pattern itself which are all on an abstract level. Implementation design patterns will be developed in the next step.

The approach aims at producing a pattern map to show relationships between all the produced patterns. In order to allow the patterns to be interrelated, the following five types of relationships between patterns are identified in the pattern-based approach:

- *extends*: when a pattern completely refines another pattern;
- *isPartOf* : when a pattern refines a part of another pattern;
- *complementsWith*: when a pattern is strongly recommended to be used with another pattern;
- *competesWith*: when two patterns provide two mutually exclusive solutions;
- *isApplicableTo*: when a pattern can be applicable to a part of the design solution provided by another pattern.

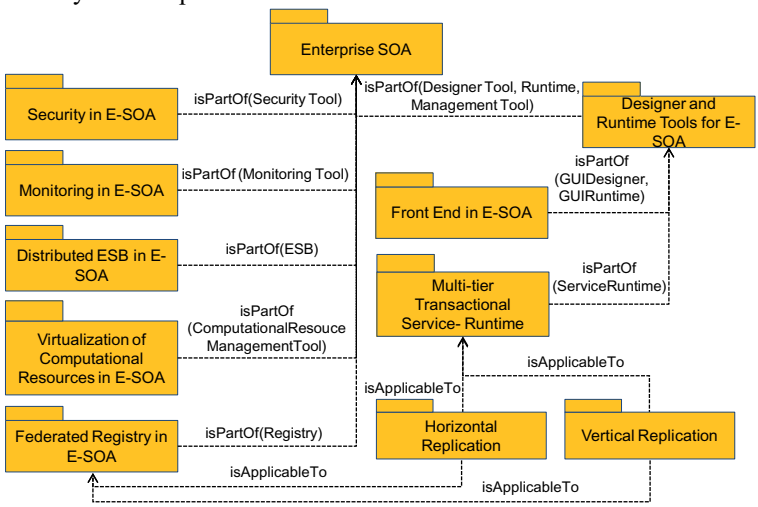

Figure 1. Excerpt of the ESOA system of patterns

Figure 1 shows an excerpt of the ESOA system of patterns[3] containing the ESOA pattern itself and two levels of abstract design patterns that refine the abstract components described by the solution proposed in the ESOA top-level pattern. The *Distributed ESB in E-SOA* pattern for example has the dependency *isPartOf* towards the *Enterprise SOA* top-level pattern. The relationship type is annotated with a list of strings in brackets which refer to the components of the higher-level pattern that are affected by the refining pattern. In this case the need to include an Enterprise Service

[3] The complete system of patterns can be found on http://www.nexof-ra.eu/?q=node/526

Bus (ESB) in an Enterprise SOA is addressed by a component *ESB* in the ESOA pattern for which a more concrete solution with certain architectural choices is specified in the refining pattern. Due to the *competesWith* relationship the system of patterns could also contain an alternative refinement of the *ESB* component that takes different, conflicting architectural choices which might apply in a different problem space. The two patterns *Horizontal Replication* and *Vertical Replication* are connected to the *Federated Registry in E-SOA* and the *Multi-tier Transactional Service-Runtime* patterns with an *isApplicableTo* relationship. The two replication patterns deal with the crosscutting issues of high availability and scalability by replicating certain components of the derived system. Thus, they can be applied in different solution contexts and not just for one problem space. An important part of these patterns is the assumption description, that specifies the join points in terms of components, relationships or certain characteristics of the architectural solution at which the replication can be applied. The result of the ongoing work will be a pattern map for ESOA that shows relationships between all the produced patterns and thus allows the derivation of a comprehensive, architectural solution.

4. The Pattern-Based Reference Architecture

In order to meet the implications discussed in Section 3.1 specifying solely a map of patterns is not sufficient for the creation of a reusable reference architecture for SBSs. Instead, a structure that is composed of several elements has been defined for the pattern-based reference architecture that provides a sound method for systematic reuse of architectural knowledge in order to derive new architectures for SBSs in certain context settings. This structure is described in the following section in order to explain how it is used to construct the pattern-based reference architecture as well as to describe how it supports the derivation of specific architectures.

4.1. The Structure of the Reference Architecture

The pattern-based reference architecture is composed of several parts, capturing the information necessary to design service-oriented systems (see Figure 2). The main constituents of the pattern-based reference architecture are the following three elements:

The guidelines and principles: This captures on the one hand the principle underlying the construction of the framework as well as the set of reference properties associated with each of the components and patterns in the reference architecture. Capturing this knowledge explicitly allows the pattern-based reference architecture to be an evolvable, dynamic reference architecture that can be continuously created by an open community in order to cope with new upcoming trends in the Future Internet.

On the other hands this part captures the guidelines used to instantiate a specific system architecture according to its requirements. Since the reference architecture will provide a huge set of architectural solutions and information it is crucial to have sound methodology that supports system architects during the derivation of actual architectures for a specific context. This can for example be a decision tree for a certain part of a system of patterns that helps the architect to evaluate which patterns are the appropriate ones for a certain context considering the different tradeoffs of the quality attributes that are associated with them.

The reference architecture model: This is the conceptual model describing the essential entities that constitute SBSs as well as the relationships between these entities and the key elements in the context. While the systems of patterns only specify the internal characteristics of the architecture for a service-based system the model also considers the applications and services that can be deployed on top of a platform realizing a derived architecture. Thus, the model tries to capture the whole SBS. Differentiating between the platform and the whole system is crucial in order to understand how the platform realizing a derived architecture is used by its surrounding. Using this knowledge, appropriate architectural solutions can be provided by the reference architecture. The model provides several different views focusing on different aspects of SBSs. These views follow the well-know approach of separating structure, behavior and functionality (cf. [12]). Following the classification of reference architectures by Beneken [8] the functional decomposition thus can be seen as what is named a functional reference architecture. Furthermore, part of it can be seen as an architecture overview of a logical reference architecture.

In addition to the different model views, this section of the reference architecture contains also a glossary, which defines the terms used across the whole pattern-based reference architecture. Thus, the reference model is part of the reference architecture and does not accompany it as an external document as proposed by Vogel et al. [9] or Bass et al. [10].

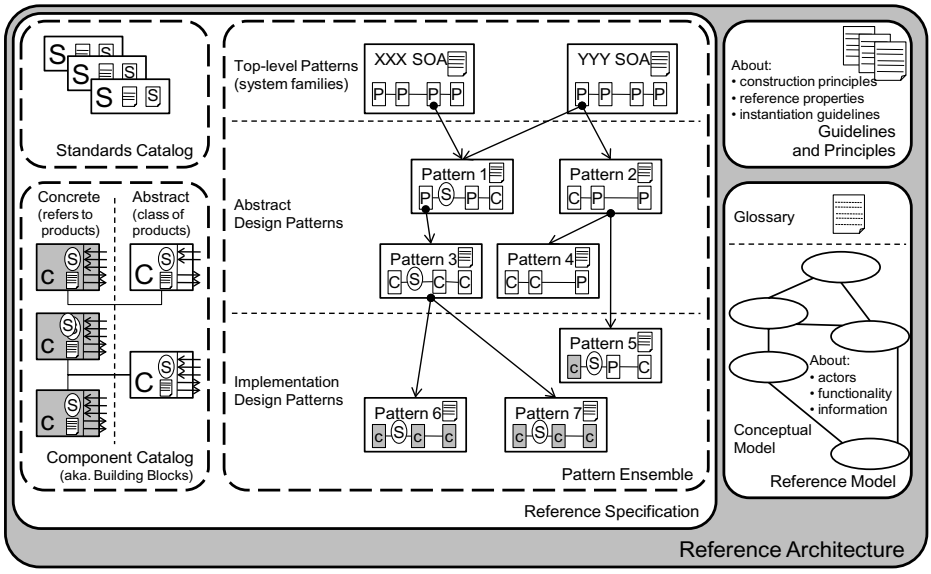

Figure 2. Structure of the pattern-based reference architecture

The reference architecture specification: This contains three collections: **The standards catalogue** describes the standards referred to in the reference architecture. Each standard is linked to the relevant elements of the guidelines and principles as well as to the concepts it addresses.

The level of granularity considered in the reference architecture is that of components, which roughly correspond to coherent sets of functionality delivered as software products or software components, which can be configured separately. **The component catalogue** groups both, abstract descriptions of components (e.g. an UDDI

registry) as well as product or software-based components (e.g. the jUDDI library). Each description refers to the standards it implements, the concepts it addresses as well as its behavioral characteristics.

The System of Pattern, as described above, represents the actionable part of the reference architecture. The specified patterns define various ways of realizing some functionality by associating components and other patterns in a defined manner. The architecture specification includes the three presented types of patterns: top-level patterns, abstract design patterns, and implementation design patterns Relationships between patterns are explicitly described. Each pattern description also refers to the standards it implements, the concepts it addresses, as well as its behavioral characteristics.

To better support the production of a set of inter related patterns, a top-down production process has been adopted for the pattern-based reference architecture. Starting from the production of top-level patterns, i.e. the most general and abstract patterns, other patterns are produced with respect to other already committed patterns. This way, the problem they address and the context where they are applicable are clearly and well-defined. Taking the ESOA family (see figure 1) into account, the *Distributed ESB in E-SOA* pattern for example can only be specified after the *Enterprise SOA* top-level pattern has been created and the use of the ESB component as well as its relationships to other components are clearly defined and thus can be refined in a separate pattern. This approach simplifies the verification of the consistency of the overall set of patterns and makes it more controllable.

The top-level patterns play a particular role in the application of the design methodology as they define different classes (or families) of SBSs that will be implemented. The principle of independence from the application domains means that the pattern-based reference architecture must be useable to instantiate compliant systems for many different application domains, such as enterprise systems, manufacturing systems, real-time systems, sensor networks, and automotive communications. In essence, the properties of SBSs used for each of these applications is different enough from the others that it constitutes a system type on its own rather than a variant in a common system family. However, once the system family (or top level pattern) has been identified, its association with abstract and implementation design patterns follows a strict structure of dependencies and refinements.

As mentioned above, the pattern-based reference architecture does not distinguish between the actual architecture description and the reference model but includes both of it. Thus, the section reference architecture specification in the presented structure constitutes the part that is traditionally referred to as reference architecture in literature. The different types of patterns and different levels of abstraction that are covered by them provide all the information that should be given in a logical as well as a technical reference architecture. The component and the standards catalogue accompany the system of patterns to make the patterns directly reusable adhering to the construction kit concept.

4.2. Application of the Reference Architecture

As identified by Fettke et al. [11] there are two processes that need to be distinguished: the construction of the reference model and the construction of company-specific information models based on these reference models. This differentiation can also be made for the pattern-based reference architecture. On the one hand it is important to

have a sound method as presented in the previous section describing how the reference architecture is build to allow the evolution of it in the future. The described approach tries to foster an adaptable and dynamic evolvable reference architecture that can cope with new trends beyond the NEXOF-RA project to provide a framework for integrating experiences and research results.

On the other hand, the process of adopting the reference architecture to derive specific architectures needs to be addressed by providing sufficient guidance. As described as part of the guidelines, the pattern-based reference architecture will come with a methodology that support system architects during the derivation of concrete architecture instances. The combination of all elements defined as part of the pattern-based reference architecture allows deriving specific instances of a software architecture for SBSs that adhere to specific requirements. The principles and guidelines provide guidance for the whole process and especially provide a starting point by describing how the requirements can be mapped towards the reference architecture.

Once a system family has been identified for the system under development, a functional decomposition should be performed based on the requirements that apply to the specific context and the different model views. These functionalities should be mapped to the functional decomposition that is provided by the reference architecture model. Such a functionality mapping is a solid baseline to actually derive the architecture instance since the various elements that are part of the pattern-based reference architecture are all interlinked. The pattern descriptions for example, map functionalities that are provided via interfaces by the specified architectural solution to the functionalities described in the model. Together with the instantiation guidelines the reference-architecture can be tailored to the specific context.

Starting from the top-level pattern of the identified system family, for all selected functionalities, the according abstract patterns can be selected following the path through the pattern map down to the implementation patterns. The functional decomposition of the model comprises, for example, functionalities addressing the issue of discovering services. More specifically it distinguishes between the functionality to search a service and the functionality to browse for a service. The ESOA top-level pattern specifies a functionality “find” that is mapped to the discovery functionality and that is related to the component *Registry* within the ESOA pattern. Within the ESOA family a pattern *Service Discovery* is specified that refines the *Registry* component. Thus, the architectural solution provided by this pattern should be included for an architecture instance that includes the discovery functionality in its functional decomposition. For different types of discovery different abstract design patterns refining the *Service Discovery* pattern are specified, e.g. the *Template-based Discovery* and the *Multi-Phase Discovery* abstract design patterns.

Considering the dependencies between the patterns, the appropriate patterns can be selected and combined on in order to describe the system architecture for a specific context. The final result will be a technical architecture that is accompanied by implementation components provided in the components catalog. The ideal scenario is that there will be enough implementation components so that the derived architecture already comes with an interoperable implementation. During the derivation, the interoperability of specific combinations of the different components should be guaranteed by interoperability levels specified in the guidelines.

5. Conclusion and Future Work

This paper presented a pattern-based approach towards the definition of a reference architecture for SBSs. This approach constitutes a reasonable solution that allows the consideration of the various types of SBSs that have been identified as relevant in the Future Internet till now. The system of patterns that has been presented allows addressing the different characteristics and requirements of the different types of SBSs that should be considered. Furthermore, the approach is dynamically extensible in order to integrate patterns that provide solutions for problems that have been identified for new trends and thus promises to become a long living framework for the derivation of SOAs of any kind. The future work towards this system of patterns will mainly deal with the definition of pattern families describing the current trends such as the Internet of Services and the Internet of Things. Furthermore, the definition of instantiation guidelines is crucial in order to make this reference architecture a useable and thus valuable framework for the derivation of specific well-defined architectures that will constitute parts of the Future Internet.

6. Acknowledgements

Research leading to these results has received funding from the EC's Seventh Framework Programme FP7/2007-2013 under grant agreement 216446 (NEXOF-RA). We cordially thank all NEXOF-RA members, Nelufar Ulfat-Bunyadi and Marian Daun for fruitful discussions and insightful comments that improved this paper.

References

[1] G. Tselentis, J. Domingue, A. Galis, A. Gavras, D. Hausheer, S. Krco, V. Lotz, T. Zahariadis, *Towards the Future Internet – A European Research Perspective*, Future Internet Assembly May 2009., Prague, IOS Press, 2009.

[2] NESSI, *NESSI Strategic Research Agenda. NESSI Research Priorities for FP, Public Vol. 3.2 Revision 2.0*, 10. May 2009.

[3] C. Alexander, S. Ishikawa, M. Silverstein, M. Jacobson, I. Fiksdahl-King, S. Angel, *A Pattern Language*, Oxford, University Press, New York, 1977.

[4] E. Gamma, R. Helm, R. Johnson, J. M. Vlissides, *Design Patterns: Elements of Reusable Object-Oriented Software*, Addison-Wesley Professional, illustrated edition, 1994.

[5] F. Buschmann, R. Meunier, H. Rohnert, P. Sommerlad, M. Stal, *Pattern-Oriented Software Architecture, A System of Patterns, Volume 1*. John Wiley & Sons Ltd, 1996.

[6] F. Buschmann, K. Henney, D. C. Schmidt, *Pattern-Oriented Software Architecture, On Patterns and Pattern Languages, Volume 5*, John Wiley & Sons Ltd, 2007.

[7] M. Jazayeri, A. Ran, F. van der Linden, *Software Architecture for Product Families: Principles and Practice*, Addison-Wesley Longman Publishing Co. Inc., 2000.

[8] G. Beneken, *Referenzarchiteckturen*, In: R. Reussner, W. Hasselbring (Ed.): Handbuch der Softwarearchitektur, dpunkt.verlag, Heidelberg, 2006 (in German).

[9] O. Vogel, I. Arnold, A. Chughtai, E. Ihler, U. Mehlig, T.Kehrer, U. Zdun, *Software-Architektur, Grundlagen – Konzepte – Praxis*, 2. Auflage, Spektrum Akademischer Verlag, 2009 (in German).

[10] L. Bass, P. Clements, R. Kazman, *Software Architecture in Practice*, SEI Series in Software Engineering, Addison-Wesley Longman, 2nd edition, Amsterdam, 2003.

[11] P. Fettke, P. Loos, *Methoden zur Wiederverwendung von Referenzmodellen – Übersicht und Taxonomi,* In: J. Becker; R. Knackstedt (Ed.): Referenzmodellierung 2002 - Methoden - Modelle – Erfahrungen, Pages 9-33, 2002 (in German).

[12] K. Pohl; Requirements Engineering: Fundamentals, Principles, and Techniques, Springer, to appear in 2010.