From i* Models to Service Oriented Architecture Models

Carlos Becerra[2,3], Xavier Franch[1] and Hernán Astudillo[2]

[1] Universitat Politècnica de Catalunya (UPC), C. Jordi Girona, 1-3 (Campus Nord, C6)
E-08034 Barcelona, Spain

[2] Universidad Técnica Federico Santa María, Avda. España 1680, Valparaíso, Chile

[3] Universidad de Valparaíso, Avda. Gran Bretaña 1091, Valparaíso, Chile

Abstract. Requirements engineering and architectural design are key activities for successful development of software systems. Specifically in the service-oriented development systems there is a gap between the requirements description and architecture design and assessment. This article presents a systematic process for systematically deriving service-oriented architecture from goal-oriented models. This process allows generate candidate architectures based on i* models and helps architects to select a solution using services oriented patterns for both services and components levels. The process is exemplified by applying it in a synthesis metadata and assembly learning objects system.

1 Introduction

Service-oriented architecture (SOA) is a flexible set of design principles used during the phases of systems development and integration [5]. A deployed service or architecture provides a loosely-integrated suite of services that can be used within multiple business domains. SOA defines how to integrate widely disparate applications for a world that is Web-based and uses multiple implementation platforms. Rather than defining an API, SOA defines the interface in terms of protocols and functionality.

One of the main problems facing architects of service-oriented systems is the gap between requirements description and architecture design and assessment.

This article presents a systematic process for deriving and evaluating service-oriented architectures from goal-oriented models. This process generates candidate architectures from i* [20] models and helps architects to select a solution, with the SOA patterns using. The i* models are used because: facilitates reasoning about the purpose of a proposed solution; i* models can be analyzed to demonstrate which goals realize other goals and which goals conflict or negatively contribute to other goals; demonstrates the contribution of the proposed and designed solution to the actual need [22].

The article is structured as follows: Section 2 presents related work; Section 2.2 describes the service oriented approach based on i*; Section 3 describes the service oriented architecture representation; Section 4 presents the Learning Objects (LOs) case study; Section 5 describes the service-oriented architecture generation process; and Section 6 summarizes and concludes.

Becerra C., Franch X. and Astudillo H. (2010).
From i* Models to Service Oriented Architecture Models.
In *Proceedings of the 4th International Workshop on Architectures, Concepts and Technologies for Service Oriented Computing*, pages 102-113
DOI: 10.5220/0003051101020113

2 Related Work

2.1 Requirements to Architectural Design

Several authors have proposed systematic approaches to obtain an architectural design from requirements description. Liu and Yu [9] proposed to explore the combined use of goal-oriented and scenario-based models during architectural design; the Goal-oriented Requirement Language (GRL) supports goal and agent-oriented modeling and reasoning, and the architectural design process; and Use Case Maps (UCM) are used to express the architectural design. The combined use of GRL and UCM enables the description of both functional and non-functional requirements, both abstract requirements and concrete system architectural models, both intentional strategic design rationales and non-intentional details of temporal features.

Chung et al. [10] proposed the NFR Framework, which uses Non-Functional Requirements (NFRs) as goals to systematically guide selection among architectural design alternatives; during the architectural design process, goals are decomposed, design alternatives are analyzed with respect to their tradeoffs, design decisions are made rationalized, and goal achievement is evaluated.

Brandozzi and Perry [11] proposed the use of Preskriptor, a prescriptive architectural specification language, and of its associated process, the Preskriptor process. Architectural prescriptions consist of the specification of the system's basic topology, constraints associated with it, and its components and interactions. The Preskriptor process provides a systematic way to satisfy both the functional and non-functional requirements from the problem domain, as well as to integrate architectural structures from well known solution domains.

Van Lamsweerde [12] presented a systematic incremental approach to deriving software architecture from system goals; it is grounded on the KAOS goal-oriented method for requirements engineering, with the intent of exploring the virtues of goal orientation for constructive guidance of software architects in their design task. It mixes qualitative and formal reasoning towards building software architectures that meet both functional and non-functional requirements.

Lucena et al. [13] presented an approach based on model transformations to generate architectural models from requirements models. The source and target languages are respectively the i* modeling language and Acme architectural description language [21]. Gross and Yu [14] proposed a systematic treatment of NFRs in descriptions of patterns and when applying patterns during design. The approach organizes, analyzes and refines non-functional requirements, and provides guidance and reasoning support when applying patterns during the design of a software system.

Grau and Franch [2] explored the suitability of the i* goal-oriented approach for representing software architectures. For doing so, they compared i*'s representation concepts against those representable in common Architecture Description Languages and defined some criteria to close the gap among these representations. They clarified the use of the i* constructs for modeling components and connectors: actors and dependencies provide an architecture-oriented semantics to help the process; added the notions of role, position and agent models in order to help traceability of the architectural representation; proposed adding of attributes to actors and model dependencies

to store information for later analysis; and suggested the use of structural metrics to analyze the properties of the final system.

All of these approaches offer systematic processes to derive requirements from architectural designs, but are not appropriate for service-orientation, because they do not provide guidelines to describe basic structures or interfaces between services. Several SOA specific characteristics demand a special approach to map requirements to architectural design alternatives, namely: 1)reuse, granularity, modularity, composability, componentization and interoperability; 2) standards-compliance (both common and industry-specific); 3) Services identification and categorization, provisioning and delivery, and monitoring and tracking. To our knowledge, only Estrada [1] has done so, proposing to address the enterprise modeling activity using i*. Estrada's [1] approach is based on using business services as building blocks for encapsulating organizational behaviors, and proposes a specific business modeling method in accordance with the concept of business service. The use of services as building blocks enables the analyst to represent new business functionalities by composing models of existing services. He proposed starting activity elicitation, the actual implementations of the services offered and requested by the analyzed enterprise, are used as basis to play a very relevant role in the discover process, and for a formal definition of the basic concepts and process design SOAs. Unfortunately, this proposal only went so far as business services, and said nothing about architectural components and connectors.

2.2 Estrada's Approach Service-orientation from i*

Estrada [1] aimed to define service-oriented architectures that address the complexity of large i* models in real-life cases. The proposed architecture distinguishes three abstractions levels (services, process and protocols) and a methodological approach to align the business models produced at these abstraction levels.

The approach includes : a) a conceptual modeling language, based on i*, which defines the modeling concepts and their corresponding relationships; b) a service-oriented architecture specific for the i* models that define the service components and the modeling diagrams. c) a business modeling method to represent services at the organizational level.

The key idea of the approach is to used business services as building blocks that encapsulate internal and social behaviors. Complementary models allow to reify the abstract concept of service to low level descriptions of its implementation.

The business service architecture is descrited by three complementary models (see Figure 1) that offer a view of what an enterprises offers to its environment and what enterprise obtains in return:

- **Global Model.** The organizational modeling process starts with the definition of a high-level view of the services offered and used by the enterprise. The global model permits the representation of the business services and the actor that plays the role of requester and provider. In this model are defined basic and compound services.
- **Process Model.** Once business services have been elicited, they must be decomposed into a set of concrete processes that perform them. This is done with a process model that represents the functional abstractions of the business process for a

specific service; this model provides the mechanisms required to describe the flow of multiple processes.

– **Protocol Model.** Finally, the semantics of the protocols and transactions of each business process is represented in an isolated diagram using the i* conceptual constructs. This model provides a description of a set of structured and associated activities that produce a specific result or product for a business service.

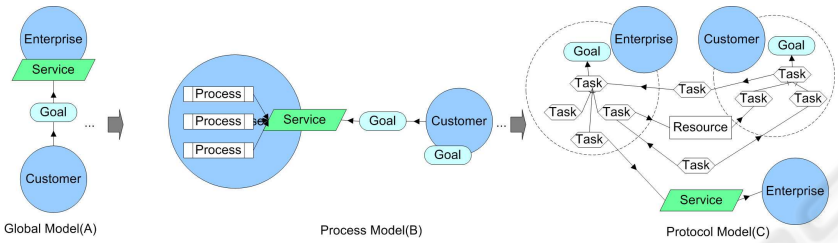

Fig. 1. A Service Oriented Approach for the i*.

The proposed approach enables the analyst to reuse the definition of protocols by isolating the description of the processes in separate diagrams. In this way, the process model represents a view of the processes needed to satisfy a service but without giving details of its implementation. Each business process is detailed through a business protocol. The detailed description of the protocols is given in the protocol model.

3 Representing Service-oriented Software Architectures

Mapping requirements to architectural design demands formalized architecture model as target, must include the notions of services, components and interfaces at different abstractions levels.

The i*-SOA Process is based on previous work by Grau and Franch [2] that defined several intentional component abstraction levels, for both services and components:

– **Service:** a set of related software functionality and the policies that control their usage. A service is accessible over standard communication protocols independent of platforms and programming languages.
– **Service Capabilities:** the operations set defined for each service [5] independently of their implementation. Therefore, this notion is especially useful during service modeling stages when the physical design of a service has not yet been determined
– **Service Components:** represents a specific component that can be integrated into the service, to implement a capability.

Connectors are described according to their abstraction level; the following types are proposed:

– **Intentional Relationships:** involve human or organizational actors and are present in the requirements models; they represent the intentional needs of the actors upon the system:
  • ***Goal Dependencies*:** functional requirement over the system.

  - ***Resource Dependencies*:** flow of concepts, or a concept relevant to the domain that does not physically exist.
- **Architectural Relationships:** occur among service components or services, as follows:
  - ***Service Interfaces*:** describe relationships among services. The dependencies definition encapsulates (hides) the deployment properties, making it vendor-programming-language and technology-independent. Service interfaces are described with Web Services Description Language (WSDL) [7].
  - ***Service Component Interfaces*:** describe components relationship within a service. they are described with the notation proposed by Han [8].

Since a pattern services concept is required to apply this in different abstraction levels, Erl's [5] set of patterns is used:

- **Services Design Patterns:** functional service contexts are defined and used to organize available service logic. Within technology-independent contexts, service logic is further partitioned into individual capabilities.
- **Composition Design Patterns:** provide the means to assemble and compose together the service logic that is successfully decomposed, partitioned, and streamlined via the service definition patterns.

Based on these definitions, the i*-SOA Process models the architecture at two different levels:

- **Service Pattern View:** In this model we apply service-oriented design patterns to describe the system architecture. There are two model sub-views:
  - **Service Design Pattern View:** Shows the structure of components and connectors for each service, based on services design patterns (e.g. redundant implementation, service data replication, message screening, etc).
  - **Composition Design Pattern View:** Shows the structure and the dependencies of services that form the system under development (e.g. service messaging, service agent, asynchronous querying, etc.).
- **Services Component View:** States the different components that exist in the service architecture (i.e. specific software component that can be integrated into the service architecture and fulfill with de capabilities). This model represents the dependencies among components within a service.

4 Introducing the Case Study

The approach reusable learning content, by combining Learning Objects (LOs) [6] has emerged in educational technology and computer science research. The approach associated with the LOs delivery rigor to the educational materials development, making the content cheaper to obtain and easy to reuse. LO are educational resources designed to generate and support learning experiences. One of the main activities to be developed in this area is to prepare courses, programs and activities based on these LO. According to this idea LO can be used by different instructors and each instructor can be reused in different learning materials.

The i*-SOA Process approach has been tried and evaluated with a Learning Object (LO) management system. The original motivation to the case study is the community need for services to improve existing LO descriptions [18, 19] and generate LO assemblies automatically. Currently, teachers and trainers have a large amount of resources (digital or not) to prepare educational materials, update their content and develop educational activities. The evolution of content distribution models from a centralized topology toward a decentralized and distributed one, has led to a scheme in which digital resources are widely and freely available. This wealth, rather than an asset, can be disadvantageous, since it adds a complexity level for users when discriminating good quality and relevant resources for specific applications.

Using LO requires collecting related information, enabling search, index and reuse. The main problem is associated with the information that user finds about a LO, which is often imperfect (imprecise, incomplete and unreliable). since many LO are not clearly classified for specific domains, search results are too general and with many possible answers list, which is not practical for users.

Thus, there are two problems to solve and whose solutions must be integrated:

- Automatically generate LO assemblies (e.g. presentations, courses, classes) from simple resources, via aggregation or composition and considering imperfect information.
- Improving LO Descriptions, gathering and synthesizing metadata from different sources.

This LO management system will be developed based on a service-oriented architecture, making available as web services the algorithms that solve the problems of LOs generation and assembly.

5 Goal Oriented Models to Service-oriented Architectural Design Process

The i*-SOA Process extends the approach by Estrada [1] described in Section 2.2. The main objective is to derive architecture at implementation level using additional model called Deployment Model. The i*-SOA Process original stages are also improved to give more semantic to dependencies intra- and inter- business services and processes. The i*-SOA Process generates alternative architectures that meet user requirements.

The method has been structured into four main activities that may iterate or intertwine as needed (see Figure 2). Section 5.1 explain the alternative SOA architectures generation process using i*-based models.

5.1 Defining the Global Model

Two complementary views of the service global model have been generated at this first phase (A).

- Abstract view of the global model: focused on representing a simple view of the offered business services (see [1]).

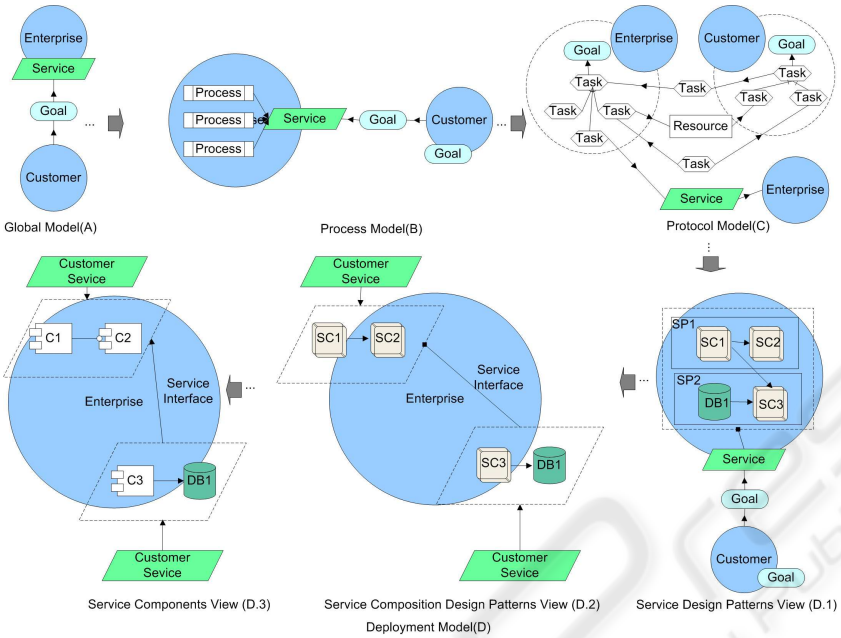

Fig. 2. The i*-SOA Process.

– Detailed view of the global model: focused on detailing the goals that are satisfied by the offered business services.

The Detailed view introduces the dependency relationships among services; specifically intentional dependencies (goal or resource dependencies) between basic services.

Figure 3 exemplifies the Global Model Detail View for the case study LO System, the main services associated with the LO Management System are:

– Learning Objects Management Service: creates new LOs descriptions from experts intentionally categorical metadata; it also allows search, update and delete of existing LOs.
– Metadata Retrieval Service: retrieves the LO descriptions dataset, to generate the initial database for the expert community.
– Metadata Synthesis Service: synthesizes and improves the LO metadata using several evidence sources.
– Learning Objects Assemblies Generation Service: using the learning objectives description provided by teachers, this service generates candidates LO assemblies that meet the requested learning objectives.

The dependencies among basic services are represented for the LO Metadata Resource Dependency and Querying LO Databases Goal Dependency.

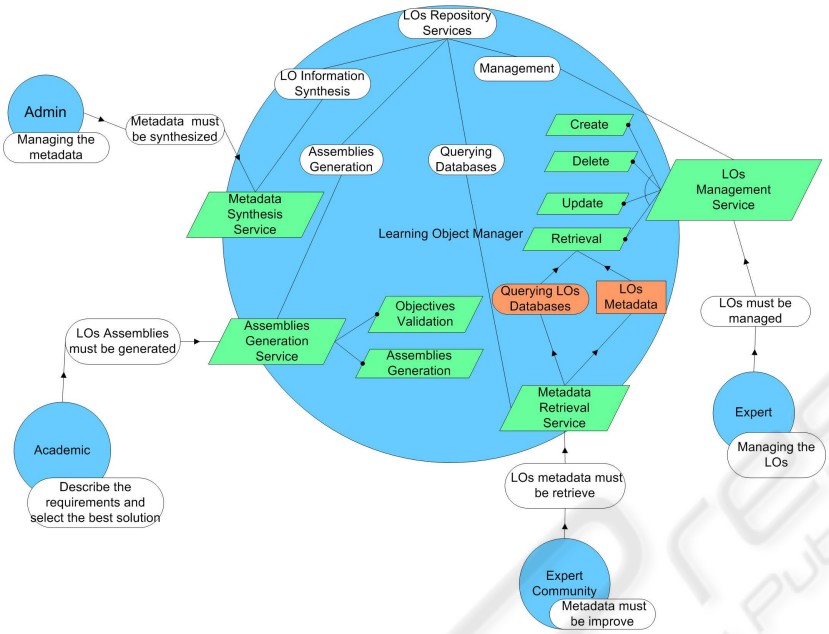

Fig. 3. Global Model Detail View.

5.2 Defining the Process Model

For each service a process model using the approach proposed in [1]. The i*-SOA Process adds the notion of dependency among processes, making necessary to specify the dependency flow and to describe the resources dependencies (resources or information). For each milestone present among processes (if required) we must specify the resource or information which helps to achieve that relationship. Figure 4 shows a LO Management Service Process Model (the resources dependencies among services processes are represented for the LO Metadata and New LO Metadata resources dependencies).

5.3 Defining the Protocol Model

The protocol model is generated based on the same process specified in [1]. Figure 5 shows a LOs Managemet Service Protocol Model.

5.4 Defining the Deployment Model

The method to define the deployment model has three sub-phases:

D.1: For each service identified in phase A:

– Based on the Process Model, identify the service design patterns (e.g. Figure 6 shows Contract Centralization, Contract Desnormalization, Concurrent Contract, Service Faade and Agnostic Capability Patterns applied in the Assemblies Generation Service) that fit the processes. For each pattern identified in the service, are specified the service components.

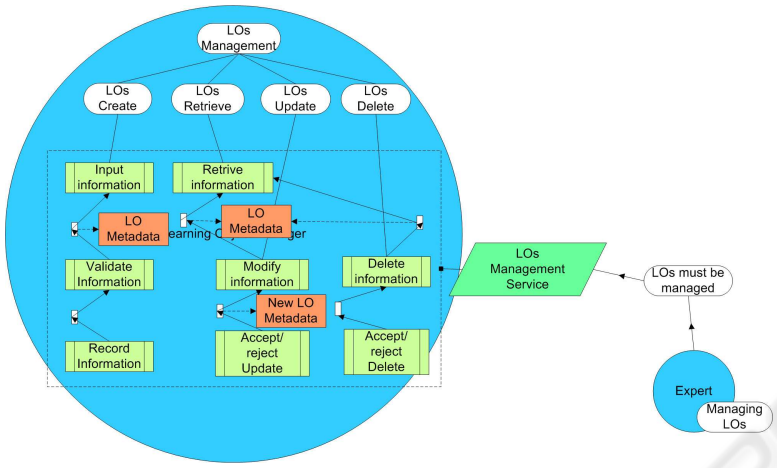

Fig. 4. LOs Management Service Process Model.

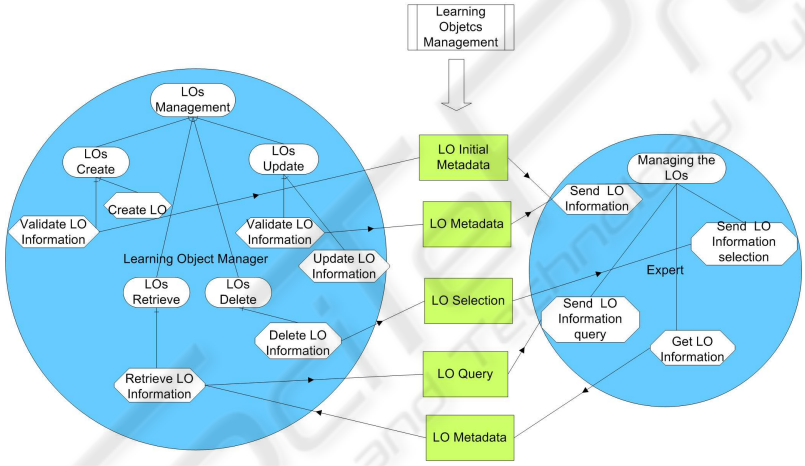

Fig. 5. LOs Management Service Protocol Model.

– For each service component specify the operations (capabilities) and the service component interfaces, which are obtained from the current Process Model activities and dependencies. Service interfaces among components are described using the notation defined in Section 3.

Identifying this pattern yields the Service Design Pattern View, which contains the service components, service components interfaces and services capabilities description for each pattern. Figure 6 shows Service Design Pattern View for one service in the running example.

D.2: Services are joined to generate the complete system architecture:

– From Service Design Pattern Views, apply service composition design patterns and structure the system, at the level of its services, services customers and services

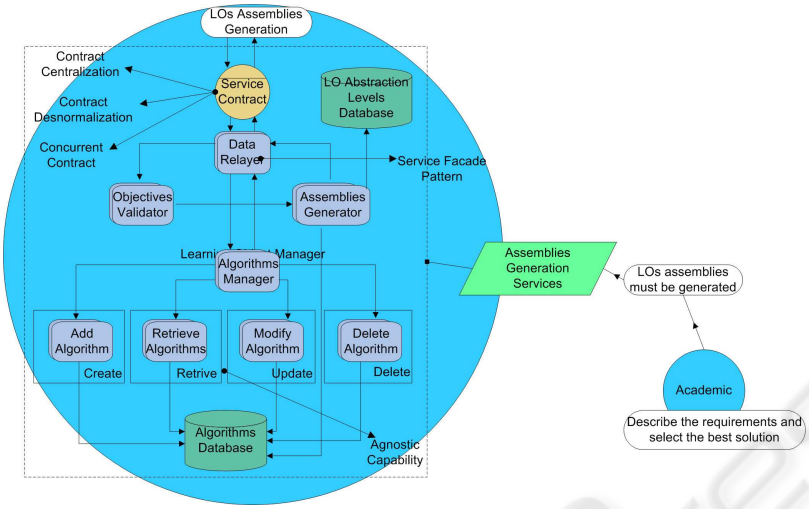

Fig. 6. LOs Assemblies Generation Service Design Pattern View.

interfaces. The interfaces among services and services customers are taken from the Protocol Model described for each service. The system service general structure and interfaces among services are taken from the Global Model. Services interfaces are described using the notation defined in Section 3.

– This sub-phase yields the Service Composition Design Pattern View. Figure 7 shows the LOs Service Composition Design Pattern View for the running example.

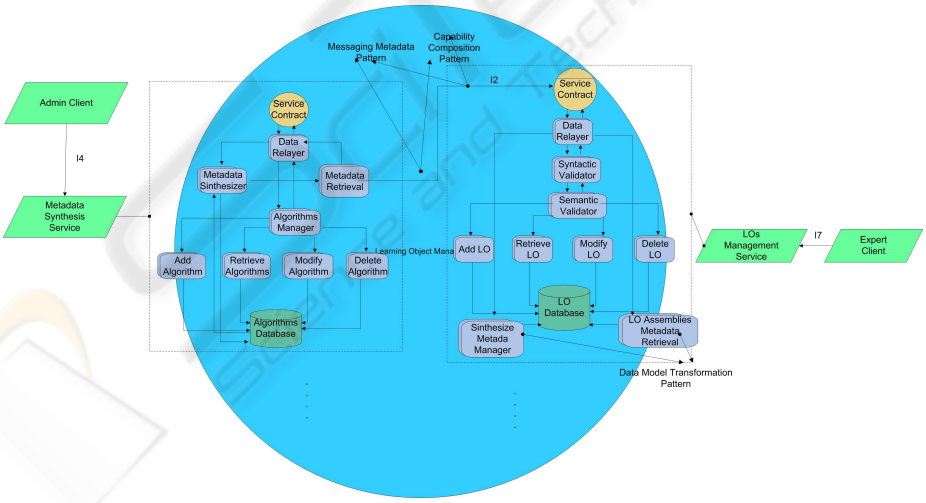

Fig. 7. LOs Service Composition Design Pattern View.

D.3: The services pattern description to specific components that implement each services capability and their interfaces. This yields the Services Component View.

6 Conclusions and Future Work

In this paper was proposed a systematic process for deriving and evaluating service-oriented architecture from goal-oriented models. This process allows to generate candidate architectures based on i* models. The main contributions are: 1) definition of basic constructs for describing a SOA architecture using i*; 2) development enables derivation of service-oriented architectures from requirements description, up to a components and connectors level; 3) description of a systematic process that applies SOA patterns in the SOA design alternatives generation.

Overall, we have proposed a systematic generation method for SOA architectures, which allows mapping requirements (specified with i*) to architectural design alternatives.

Future work will extend this proposal up to a technological solutions level, associated with the architectural design. We are also developing a method to select and evaluate SOA alternatives design, including models and metrics to generate and evaluate the solutions. We are developing automated support and/or adopting and possibly extending existing tools for this proposal, and validate the efficiency an effectiveness of this proposal with an experimental study (after and before implement an automatic support).

Acknowledgements

This work has been partially supported by the Spanish project TIN2007-64753.

References

1. Estrada, H.: “A service oriented approach for the i* framework”. Universidad Politcnica de Valencia Phd. Thesis, 2008. Thesis Director Oscar Pastor Lpez.
2. Grau, G. and Franch., X.: “On the Adequacy of i* Models for Representing and Analyzing Software Architectures”. Advances in Conceptual Modeling Foundations and Applications, 2007, pages 296-305.
3. Rud, D., Schmietendorf, A., Dumke, R.: “Product metrics for service oriented infrastructures”. In Proceedings of the 16th International Workshop on Software Measurement and DASMA Metrik Kongress (IWSM/MetriKon 2006), pp. 161-174, November 2-3, 2006, Potsdam, Germany.
4. Aier, S. and Ahrens, M., and Stutz, M., and Bub, U.: “Deriving SOA Evaluation Metrics in an Enterprise Architecture Context”. Service-Oriented Computing - ICSOC 2007 Workshops: ICSOC 2007, International Workshops, Vienna, Austria, September 17, 2007, Revised Selected Papers, 2007.
5. Erl. T.: “SOA Design Patterns”. Prentice Hall/PearsonPTR, , Upper Saddle River, NJ, USA, 2009
6. IEEE. draft standard for learning object metadata - proposed standard. Technical report, IEEE, Piscataway, 2002.
7. Web Services Description Language (WSDL) Version 2.0 Part 0: Primer, W3C Working Draft 3 August 2005, http://www.w3.org/tr/2005/wd-wsdl20-primer-20050803/

8. Han, J.: “A Comprehensive Interface Definition Framework for Software Components”. APSEC ’98: Proceedings of the Fifth Asia Pacific Software Engineering Conference 1998, IEEE Computer Society.
9. Liu, L. and Yu, E: “From Requirements to Architectural Design - Using Goals and Scenarios”. First International Workshop From Software Requirements to Architectures (STRAW 01), 2001, Toronto, Canada.
10. Chung, L., Nixon, B., and Yu E.: “Using Non-Functional Requirements to Systematically Select Among Alternatives in Architectural Design”. Proc. 1st Int. Workshop on Architectures for Software Systems, 1994, pp. 31-43.
11. Brandozzi, M., Perry, D.E.: “From goal-oriented requirements to architectural prescriptions: the preskriptor process”. Second International Software Requirements to Architectures Workshop (STRAW’03)., 2003, pp. 107-113.
12. Van Lamsweerde, A.: “From system goals to software architecture”. Formal Methods for Software Architectures, 2003, pages 25-43.
13. Lucena, M., Castro, J., Silva, C., Alencar, F., Santos, E. and Pimentel, J.: “A Model Transformation Approach to Derive Architectural Models from Goal-Oriented Requirements Models”. OTM ’09: Confederated International Workshops and Posters on On the Move to Meaningful Internet Systems: ADI, CAMS, EI2N, ISDE, IWSSA, MONET, OnToContent, ODIS, ORM, OTM Academy, SWWS, SEMELS, Beyond SAWSDL, and COMBEK 2009, Vilamoura, Portugal, pp. 370-380.
14. Gross, D., and Yu, E.: “From Non-Functional Requirements to Design through Patterns”. Requirements Engineering, Volume 6 (1), 2001, pp. 18-36.
15. Liu, Y. and Traore, I.: “Complexity Measures for Secure Service-Oriented Software Architectures”. PROMISE ’07: Third International Workshop on Predictor Models in Software Engineering, 2007.
16. Qian, K., Liu, J., and Tsui, F.: “Decoupling Metrics for Services Composition”. ICIS-COMSAR ’06: 5th IEEE/ACIS International Conference on Computer and Information Science and 1st IEEE/ACIS International Workshop on Component-Based Software Engineering, Software Architecture and Reuse, 2006, pp. 44-47.
17. Hirzalla, M., Cleland-Huang, J., Arsanjani, A.: “A Metrics Suite for Evaluating Flexibility and Complexity in Service Oriented Architectures”. ICSOC 2008 Workshops: ICSOC 2008 International Workshops, Sydney, Australia, December 1st, 2008, pp. 41-52.
18. Chan, L.M.: “Inter-Indexer Consistency in Subject Cataloging”. Information Technology and Libraries, 1989. 8(4): p. 349-358.
19. Currier, S., Barton, J., O’Beirne, R., and Ryan, B.: “Quality Assurance for Digital Learning Object Repositories”. Issues for the Metadata Creation Process. ALT-J, research in Learning Technology, 2004. 12(1): p. 6-20.
20. Mylopoulos, J., Chung, L., Yu, E.: ”From Object-Oriented to Goal-Oriented Requirements Analysis”; Commun. ACM 42(1): 31-37 (1999).
21. Garlan, D., Monroe, R. and Wile, D.: ”Acme: An Architecture Description Interchange Language”; Proceedings of CASCON97, 1997, 169–183.
22. Quartel, D.A.C., Engelsman, W., Jonkers, H., and van Sinderen, M.J. “A goal-oriented requirements modelling language for enterprise architecture”. Thirteenth IEEE International EDOC Enterprise Computing Conference, EDOC 2009, 1-4 Sep 2009, Auckland, New Zealand. pp. 3-13. IEEE Computer Society Press.