Doctoral School on Engineering Sciences
Università Politecnica delle Marche

Extended summary

Formal Methods for Service Oriented Software Engineering

Curriculum: Ingegneria Informatica, Gestionale e dell'Automazione

Author
Emanuele Glorio

Tutor(s)
Prof. Luca Spalazzi

Date: 30-01-2013

Abstract. *Service-Oriented Computing* is becoming more and more important. The proliferation of grid and cloud computing is increasing this trend. As a result, more companies than ever before are exposing their Web services on the Internet. This fact has the effect of transforming the web from a repository of data to a repository of service [1]. In this scenario, a software engineer is called to design a software taking into account the opportunity/need of reusing existing services. This requires two issues: - a software engineering methodology that starts from business goals and organization of a given company and arrives to identify which parts can be delegated to external services; - the capability of locating the "most" appropriate services. In fact, while technology and standards, such as Web services, are important, it has been widely recognized that they are not sufficient on their own. Instead, a systematic and comprehensive approach is of critical importance, taking into account the business requirements and following recommended practices. For this reason, even if there are many service-oriented methodology nowadays, *Service-Oriented Software Engineering* (SOSE) is still an open field. In this thesis we present the definition of a new SOSE methodology. As start, we use Tropos early phases because it is an agent-oriented methodology which bears particular attention to stakeholder needs and requirements analysis. Besides, Tropos was already refined in order to support web service design. We leave the first three phases unchanged (*Early and Late Requirements*, *Architectural Design*) and we focus on the final phases (*Detailed Design* and *Implementation*). In *Detailed Design* phase we propose a mapping between i* concepts and BPMN 2.0 elements in order to translate automatically the i* diagram derived from the previous phases in a workflow language. Moreover we provide formal methods and techniques to select code and services in order to reuse them inside the to-be application. Finally, in *Implementation* phase we propose a mapping BPMN - Alan (an agent-object oriented programming language) to produce automatically an executable application. We present a case study from e-commerce and we use it to show how to apply our methodology step by step.

Keywords. Formal methods, Service-oriented software engineering, Agent-oriented software engineering, Service selection

1 Problem statement and objectives

A number of SOA methodologies such as IBM RUP/SOMA, SOAF, SOUP, etc. have been proposed to ensure successful SOA development. IBM RUP/SOMA [2] is an integrated methodology developed by IBM in a will to bring unique aspects of SOMA to RUP. However, because SOMA is a proprietary methodology of IBM, its full specification is not available. Service Oriented Architecture Framework (SOAF) [3] methodology aim is to ease the service identification, definition and realization activities by combining a top-down modeling of an existing business process with a bottom-up analysis of existing applications. The methodology by Papazoglou [4] is SOA development methodology that covers a full SOA lifecycle. It is partly based on well-established development methodologies as RUP, Component-based Development and BPM. The methodology by Thomas Erl [5, 6] is a step by step guide through the two main phases: service-oriented analysis and design. Service-oriented Unified Process [7] or SOUP is a hybrid software engineering methodology that is targeted at SOA projects and is primarily based on the Rational Unified Process. In the BPMN to BPEL [8] methodology the business process is expressed in an abstract model (BPMN) and according to transformation rules it is automatically mapped to an execution language (BPEL) that can be executed by a process engine. Even considering the specific characteristics of services during design, most of the proposal methods do not provide specific approaches for the requirements engineering phase. The analyzed SOA methodologies are built upon and incorporate existing and proven techniques, notations such as OOAD (Object-Oriented Analysis and Design), CBD (Component-Based Development), BPM (Business Process Management), WSDL, BPEL, UML. However, the service paradigm introduces unique requirements that should be addressed by innovative techniques. For this reason we introduce also a requirements-driven methodology, called Tropos[citare] that comes from agent-oriented software engineering but fits for our purpose. Tropos is a well known software development methodology which is founded on the i* [9] organizational modeling framework. i* offers concepts such as actor (actor can be agents, positions or roles), as well as social dependencies among actors, including goal, softgoal, task and resource dependencies. Tropos has five phases: Early requirements, Late requirements, Architectural Design, Detailed Design, Implementation. Tropos framework has been related to different application areas, including requirements engineering, software processes, and business process reengineering. In particular has been extended also for service engineering ([10, 11, 12, 13]). In these works, however, the Tropos methodology is used for the early phases of the service design and is not complete. In fact, they only describe the business organization (intentional aspects of the problem) but do not address the workflow description. They directly derive from intentional aspects the capacities that services must have.

2 Research planning and activities

From our point of view, the current SOSE methodologies lack in several aspects, namely:

- Even if it is widely recognized the importance of deriving a SOA design from business needs, no methodology covers both preliminary and detailed design phases. Most of the proposed methodologies focus on architecture and component design (e.g. IBM RUP/SOMA, SOAF, SOUP, etc.) and do not deepen how the business and organizational model can be derived by requirements. Some methodologies (e.g. Tropos) focus on the preliminary phases but do not deepen how services can be identified, selected, and composed.
- Even if formal methods have been widely used into the web service reserach area, usually they are not integrated in a clear and well defined software engineering methodology. Most of the works usually focus on the specific problem they try to solve, e.g. selection, composition, and so on.

For these reasons we decided to create a new methodology:

- Concerning the early stages of the service candidate identification (i.e. the creation of the business model) we decided to adopt Tropos that uses i* as modelling language
- Regarding the final stage of the service candidate identification we use BPMN 2.0. We provide a mapping to switch automatically from i * to BPMN in order to create the service model from business model. We have chosen BPMN as modelling language because it is a standard as well as it is more understandable than for example UML. In this way the design validation phase continues even after the early stages where the designer uses i*. In fact, since the BPMN is understandable to the stakeholders they can continue to follow the design.
- Concerning the stage of implementation we provide the designer with the possibility of using directly a programming language (Alan) showing an automated mapping BPMN-Alan. Moreover we allow to use code and services already existing through a mechanism of automatic selection. The designer can use a high-level language such as BPMN to easily create a specification, a facsimile to what he or she wants to obtain. In this way, is provided the ability to reuse code already written (by himself/herself or by others) or invoke external services to delegate entire operations.

The main advantages of our methodology are:

- We derive a workflow from intentional aspects, and we allow the designer to modify and complete it
- We use semantic annotation to ease interoperability
- We give the opportunity to identify services and then select them automatically. We perform the identification process manually and in this work we do not examine it in depth because there are works [14, 15] that perform this automatically.
- We give the opportunity to perform service composition

3 Analysis and discussion of main results

In this Section we show in detail our methodology (see Figure 1). Since we used the first three phases of Tropos that are already formalized in [10, 11, 12], we do not explain them. Instead we go into detail of the four and five phase (*Detailed Design* and *Implementation*).

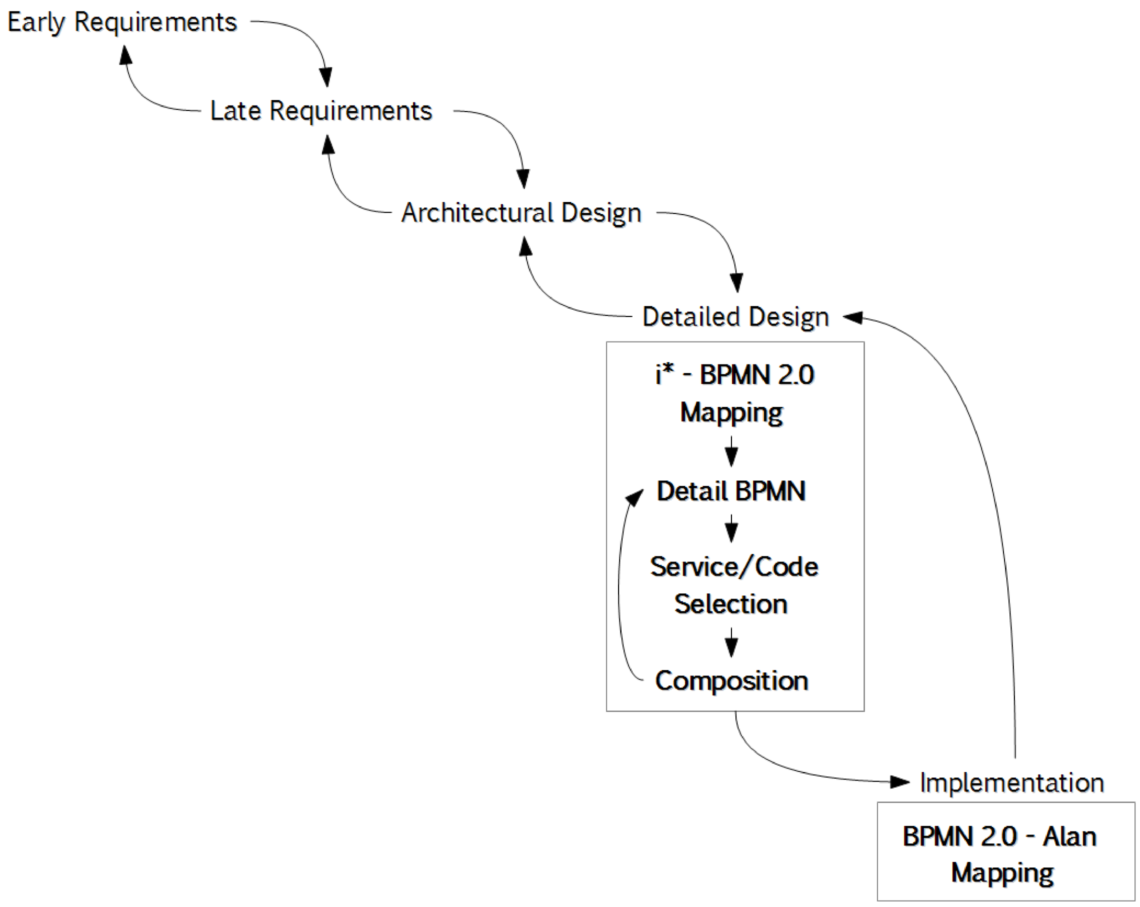

Figure 1. Life cycle of our methodology.

3.1 Detailed Design

The Detailed Design phase consists in four sub-phases: *i* to BPMN Mapping*, *BPMN Detail*, *Service & Code Selection* and lastly *Composition*.

3.1.1 i to BPMN 2.0 Mapping*

As mentioned before, we decided to use BPMN language to represent actor communications and actor behaviors at Detailed Design phase. Hence the first step of this phase is to convert the i* diagram of the previous phase into a BPMN diagram. We created a mapping between the i* constructs and the BPMN 2.0 elements, so it is possible to do the translation in a automatic way. In Table 1 we show our mapping.

3.1.2 Detail BPMN

The first automated step generates a BPMN skeleton with all the concepts and elements that are present in i* Architectural Design graph. As expected, this skeleton may be not directly executable or even directly translatable into a programming language. So, in this second step the designer has to refine the diagram in order to make it complete and eventually runnable. The operations he should perform are:

- detail the content of each plan in order to make them runnable

Table 1. i* - BPMN 2.0 Mapping

i* concept	BPMN 2.0 element
Actor	Pool
HardGoal	Start and End Event
Main HardGoal	Start and End Event inside a Lane
Plan	Sub-Process
HardGoal AND Decomposition	Parallel Gateway
HardGoal OR Decomposition	Exclusive Gateway
Task AND Decomposition	Parallel Gateway
Task OR Decomposition	Exclusive Gateway
Means-End	Sub-Process linked to Events
Resource	Data Object
Goal dependency	Message flow between Events
Task dependency	Message flow between Event and Sub-Process
Task dependency with external Role	Service Task

- finish off the rest of BPMN diagram:
  - he could transform the existing elements in more specialized elements (e.g transform a simple gateway in a event-based or databased gateway)
  - transform an AND-gateway in a sequence
  - and eventually he could add more BPMN elements like other events, tasks, gateways and messages.

One of the operations that the designer has to perform in this step is to particularize the content of each sub-process. A sub-process corresponds to an i* plan, hence it has to contain all the atomic actions required to achieve the corresponding plan. We focus in particular on this operation because it is the most important for our methodology. In fact, as shown before, our work aims at helping the designer in this particular operation providing an automated support. In our methodology the designer do not have to specify every single action that will be present in the final plan, but only the ones that are meaningful to him. In other words, the designer specifies the only things he know about how the plan should work. In this way he puts very little effort into particularizing the agent's plan, however with our system he can achieve the same result as a traditional and more laborious way. The designer starts with the almost empty sub-process created in the previous step, containing only a start event, an end event and a simple task as placeholder. Inside the sub-process he can add other BPMN elements, like gateways, tasks, events to create a model of the process he has in mind. As the designer adds elements in the sub-process, he has to semantically annotate them, but with a drag-and drop visual operation the annotation is very simple and user-friendly. All that is necessary because our framework is based on semantics to perform selection.

3.1.3 Service & Code Selection

As a result of the previous step, for each plan the designer decided to detail in BPMN we have an annotated process. For these plans, he chose to benefit of our automated selection system, instead of implement directly them. Every single process represents the designer's requirement of how the respective plan should work. Therefore, in this step we perform the actual selection process. In a few words, we provide a framework in which the annotated BPMN is translated into a temporal logic formula and used to check if there is a similar process into a repository of processes. Our methodology provides two different kinds of selection: local and remote selection. In the local selection our framework search for possible matches in a local repository of source code. This is the case when the designer want to reuse some existing code and paste it in the system instead of rewriting it. On the other hand, in the remote selection the search is done in a remote repository of web services. In this case our framework will find a web service that can be invoke remotely, with no need of copy and paste some code into the designer's code. It is up to the designer to choose which selection to use, and the choice must be made in the previous phase (i.e. *Architectural Design* phase). If the designer wants to find a remote service he has to create in the first place an external role in his system and then a Task Dependency involving this role as dependee. Consequently our mapping creates in BPMN a Service task associated with a message to a subprocess in which the designer details his requirements. After the selection is complete our framework set the parameters of the selected web service to the Service Task that is responsible to invoke it and ignore the subprocess that is no longer needed. In all the other cases (i.e. task inside the system and Task Dependency with an internal actor) our framework performs local selection.

3.1.4 Composition

In this last step we have to compose the processes (i.e. code or services) selected previously. After the selection step, our system replace the annotated BPMN used for requirement with the retrieved code or service. This is an automated process, and it is not enough. In fact, the several services obtained must be composed in order to work properly. This composition is possible manually by the designer or automated with traditional planning techniques. It is also possible that in this step we do not have any results from the previous selection process, because there are no services (or code) that fulfill his/her requirements. In this case the designer has to refine the workflow in order to change or simply relax the specifications. To do that, the designer goes back to the second step, refines the annotated BPMN and repeats the selection. This iteration ends when all the requirements are fulfilled and it is possibile to start the implementation phase.

3.2 Implementation

First of all, we want to point out that BPMN 2.0 is an executable language so, in principle, the diagram obtained from the design phase can be execute directly in a BPMN engine. Since the BPMN stays at an high level and in many cases this is not enough, it is often necessary to use a real programming language. We decided to use Alan as programming language because it is a language that combine both agent-oriented and object-oriented programming. Moreover, Alan is great to translate business and social elements to programming concepts, helping us to reduce the semantic gap between the final system and its operational environment. In Table 2 we present the mapping between BPMN elements and Alan classes. The mapping i*-Alan has already been formalized in [16] and we used it to create our translation in Alan.

Table 2. BPMN 2.0 – Alan Mapping

BPMN 2.0 element	Alan class
Pool	Alan agent
Lane	We translate only the content of the Lane and not the Lane itself
Event	Desire
Sub-Process	Plan
Data Object	Belief
Parallel gateway outside Sub-Processes	Desire ANDlist
Exclusive gateway outside Sub-Processes	Desire ORlist
Message flow between Pools	Message exchange between Alan agents
Service Task	Plan with Java code inside planbody()
Data Object Association	Belief inside Alan class
Text Annotation	Comment
Normal sequence flow of Activity	Sequential operations
Split Exclusive Gateway	if/else if/else conditional statement
Split Parallel Gateway	Threading
Join Exclusive Gateway	Conditional statement
Loop	while statement
Exception Flow	try/catch statement

4 Conclusions

In this work we have presented the definition of a new SOSE methodology. We have shown the motivation behind the creation of it and the importance of a new well defined and formally founded methodology. As start, we used Tropos early phases because it is an agent-oriented methodology which bears particular attention to stakeholder needs and requirements analysis. Besides, Tropos was already refined in order to support web service design ([10, 11, 12, 13]). Since we wanted to have a complete methodology and the previous work with Tropos were incomplete regarding the final phases, we decided to create this new methodology based on Tropos. We have left the first three phases unchanged (Early and Late Requirements, Architectural Design) and we have focused on the final phases (Detailed Design and Implementation). In Detailed Design phase we have proposed a mapping between i* concepts and BPMN 2.0 elements in order to translate automatically the i* diagram derived from the previous phases in a workflow language. Moreover we have provided formal methods and techniques to select code and services in order to reuse them inside the to-be application. Finally, in Implementation phase we have proposed a mapping BPMN - Alan to produce automatically an executable application. We have presented a case study from e-commerce and we have used it to show how to apply our methodology step by step. About future work, our methodology should be tested on a large set of real cases in order to validate it. In fact, meanwhile the selection process has been already extensively tested, the whole methodology process is new and untested. Besides, it should be created an integrated framework for our methodology, since currently there are a lot of separated tools doing everyone a part of the process.

References

[1] Qi Yu and Athman Bouguettaya, *"Guest editorial: Special section on query models and efficient selection of web services"*, Services Computing, IEEE Transactions on, vol. 3, no. 3, pp. 161 –162, july-sept. 2010.

[2] P. Kruchten, *The Rational Unified Process: An Introduction*, The Addison-Wesley Object Technology Series. Addison-Wesley, 2004.

[3] Abdelkarim Erradi, Sriram Anand, and Naveen N. Kulkarni, *"Soaf: An architectural framework for service definition and realization"*, in IEEE SCC, 2006, pp. 151–158.

[4] M.P. Papazoglou and W.J. Van Den Heuvel, *"Service-oriented design and development methodology"*, International Journal of Web Engineering and Technology, vol. 2, no. 4, pp. 412–442, 2006.

[5] Thomas Erl, *Service-Oriented Architecture: Concepts, Technology, and Design*, Prentice Hall PTR, Upper Saddle River, NJ, USA, 2005.

[6] Thomas Erl, *SOA Principles of Service Design* (The Prentice Hall Service-Oriented Computing Series from Thomas Erl), Prentice Hall PTR, Upper Saddle River, NJ, USA, 2007.

[7] Kunal Mittal, *"Service Oriented Unified Process (SOUP)"*. Available from: http://www.kunalmittal.com/html/soup.html

[8] C. Emig, J. Weisser, and S. Abeck, *"Development of soa-based software systems - an evolutionary programming approach"*, in International Conference on Internet and Web Applications and Services/Advanced International Conference on, feb. 2006, p. 182.

[9] E. Yu, *Modelling Strategic Relationships for Process Reengineering*, Ph.D. thesis, Department of Computer Science, University of Toronto, Toronto, Canada, 1995.

[10] Loris Penserini, Anna Perini, Angelo Susi, and John Mylopoulos, *"From stakeholder intentions to software agent implementations"*, in Advanced Information Systems Engineering, Eric Dubois and Klaus Pohl, Eds., vol.4001 of Lecture Notes in Computer Science, pp. 465–479. Springer Berlin Heidelberg, 2006.

[11] Diana Lau and John Mylopoulos, *"Designing web services with tropos"*, in Proceedings of the IEEE International Conference on Web Services, Washington, DC, USA, 2004, ICWS '04, pp. 306–, IEEE Computer Society.

[12] Amy Lo and Eric Yu, *"From business models to service-oriented design: a reference catalog approach"*, in Proceedings of the 26th international conference on Conceptual modeling, Berlin, Heidelberg, 2007, ER'07, pp. 87–101, Springer-Verlag.

[13] Marco Aiello and Paolo Giorgini, *"Applying the tropos methodology for analysing web services requirements and reasoning about qualities of services"*, 2004.

[14] Devis Bianchini, Cinzia Cappiello, Valeria De Antonellis, and Barbara Pernici, *"P2s: A methodology to enable inter-organizational process design through web services"*, in CAiSE, 2009, pp. 334–348.

[15] Devis Bianchini, Francesco Pagliarecci, and Luca Spalazzi, *"From service identification to service selection: An interleaved perspective"*, in Formal Modeling: Actors, Open Systems, Biological Systems, 2011, pp. 223–240.

[16] L. Spalazzi F. Pagliarecci, L. Penserini, *"A Goal-Oriented Framework to cope with Requirements Changes: Tropos & Alan"*, International Transaction on Systems Science and Applications, vol. 4, no. 4, pp. 367–381, December 2008.