
	<p>TIN2013-46638-C3-1-P</p> <p>15/10/2014</p>	
---	--	--

Reference: TIN2013-46638-C3-1-P

Project full title: Probabilistic graphical models for scalable data analytics

Project Acronym: PGM-SDA

Deliverable no.: D16.1

Title of the deliverable: Methodology for service oriented requirement analysis

Start date:	15/10/2014
End date:	
Author(s):	Antonio Fernández Álvarez; Irene Martínez Masegosa; Rafael Rumí Rodríguez; José del Sagrado Martínez; Antonio Salmerón Cerdán
Participant(s):	Irene Martínez Masegosa; José del Sagrado Martínez; Antonio Fernández Álvarez; Luis de la Ossa; Pablo Bermejo; Rafael Cabañas de Paz; Javier García Castellano; Manuel Gómez Olmedo
Objective:	7. Software platform development
Version:	0.1
Total number of pages:	
Start date of project:	1st January 2014. Duration: 36 months

Abstract:

Keywords list:

Contents

- 1. Motivation**
- 2. SOA basics**
- 3. SOA lifecycle**
- 4. Services**
 - 1. Classification**
 - 2. Service identification**
 - 3. Service analysis and design**
- 5. Get the requirements for a single service: Service Specification**
- 6. Methodology for service oriented requirement analysis**

Document history

<i>Version</i>	<i>Date</i>	<i>Author (Unit)</i>	<i>Description</i>
v0.1	15/10/2014	Antonio Fernández Álvarez; Irene Martínez Masegosa; Rafael Rumí Rodríguez; José del Sagrado Martínez; Antonio Salmerón Cerdán	First draft

1 Motivation

The project “**Probabilistic graphical models for scalable data analytics (PGM-SDA)**” has as one of its main objectives to produce the necessary software tools as to allow the development of applications based on a web-services architecture for PGMs. In this way, we expect to create a framework where mobile hardware devices could be used in big data contexts, as the core processing would rely on a centralized server that would process the data and run the necessary algorithms, while the mobile device would interact through the web services interface. A Service Oriented Architecture in the field of PGMs (PGM-SOA) can provide a general framework for organizing algorithms integration and serves as basis for the development of information systems for PGMs based on open platforms, data communication and software interoperability standards.

This document describes the service-oriented requirements engineering (SORE) process adopted in the PGM-SDA project. As there is no common and agreed method for conducting SORE for distributed scenarios, the PGM-SDA SORE process (based on selected methodological approaches from existing RE processes) has as main goal to identify basic services that provide basic functionality (at the level of business) for a specific problem domain (applications) conforming a Service Oriented Architecture in the field of PGMs (PGM-SOA).

The PGM-SDA SORE process is conceived as an aid to elucidate the PGM-SOA and, thus, has been tailored to the specific characteristic of the PGM-SDA project: several research groups are developing methods and algorithms, often internationally and, usually, each one of them use an own system and software for making its own developments. Because of that, this process is focused on the joint elicitation of service requirements from use case providers coming from different research groups geographically dislocated.

This is partly achieved by the development of a unified formal elicitation template for service-oriented requirements. The template also supports transparency in the overall service-oriented requirements engineering process and helps prioritize requirements and resolve potential conflicts across domains. However, non-functional requirements are out of the process scope, but are mentioned in the project memory document.

2 SOA basics

First of all, let us begin giving a definition of SOA. There are two formal definitions created, respectively by the OASIS [OAS06] group and the Open Group [OPE09].

- OASIS defines SOA [OAS06] as:

“A paradigm for organizing and utilizing distributed capabilities that may be under the control of different ownership domains. It provides a uniform means to offer, discover, interact with and use capabilities to produce desired effects consistent with measurable preconditions and expectations.”

- The Open Group defines SOA [OPE09] as:

“Service-Oriented Architecture (SOA) is an architectural style that supports service-orientation.

Service-orientation is a way of thinking in terms of services and service-based development and the outcomes of services.

A service:

- *Is a logical representation of a repeatable business activity that has a specified outcome (e.g., check customer credit, provide weather data, consolidate drilling reports)*
- *Is self-contained*
- *May be composed of other services*
- *Is a “black box” to consumers of the service”*

From the last formal definition, when we develop applications in the PGM-SDA framework, the main features of PGMs should be perceived as services in a distributed system. Thus, we consider that a *Service-oriented architecture (SOA)* [Abu08] is a distributed transaction scheme for achieving interoperability in heterogeneous distributed systems deployments, which can be viewed as an interaction between a service requester and a service provider. There are three types of roles (see Fig. 1) in SOA architecture [Jia09]:

- *service provider* publishes its own services,
- *service broker* register provider of issued service, classifies them and provides search services,
- *service requester* searches the services they need using a service agent, and then uses the service it has found.

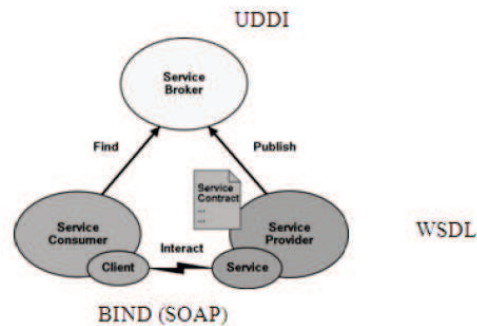


Fig. 1. SOA Architecture.

Every service has to be well defined in order to be usable by requesters/consumers. The elements of a service are:

- A *contract* which specifies what a service provider offers in order to cover the needs of a service consumer.
- An *interface* that defines how a service can be accessed and used.
- An *implementation* is the software realization of the service specification.

Consumers can access to the *contract* and *interface* of a service, whilst its *implementation* is kept hidden. The objective of consumers is to use services and they are not interested in the details of their implementations.

Following [Jus07], the *key technical concepts of SOA* are:

- *Services*. As the goal of SOA is to structure distributed systems based on the abstraction of business rules and functions, a service can be considered as an IT representations of business functionality. Externally services hide technical details and through their interfaces should be designed in such a way that business people can understand them.
- *Interoperability*. As services spread over heterogeneous systems, the goal of SOA is to connect those systems easily.
- *Loose coupling*. Loose coupling deals with the minimization of dependencies. When dependencies are minimized, modifications have minimized effects, and the systems is fault tolerant (it still runs when parts of it are broken or down). Minimizing dependencies contributes to *fault tolerance* and *flexibility*. In addition, loose coupling leads to *scalability*. Large systems tend to challenge limits. Therefore, it is important to avoid bottlenecks eluding cost for growing. One way to introduce loose coupling is to avoid introducing any more centralization than is necessary (unfortunately, you need some centralization to establish SOA because you need some common base).

In summary [Dik12], SOA tries to *decouple* (or *loose-couple*) data and logic that do not belong together through the appliance of services. This loose coupling occurs on the level of ownership, business logic, data, and deployment. The SOA bases are *services*, which are small building blocks that provide clear access to a limited set of capabilities that belong together. For a particular set of capabilities, the same service is responsible for the business logic and data consistency. If the data belonging to a service

needs to be changed, it is done through that service alone, thus enforcing a single point of access for that particular functionality and data.

3 SOA lifecycle

The SOA lifecycle, as is described by the IBM SOA Foundation [Hig05], is a process that comprises two development activities (*model* and *assemble*) and two operation activities (*deploy* and *manage*). The four phases, performed iteratively, are [Mitt06, High05]:

- *Model*. Requirements are gathered, end-to-end business processes are modeled, analyzed, designed, and then further optimized to form the future state business processes for the enterprise. This activity is in charge of business analysis and design (requirements, processes, goals, key performance indicators) and IT analysis and design (service identification and specification).
- *Assemble*. Services are implemented. The implemented services are then assembled; that is, they are discovered, choreographed and composed to implement the enterprise business processes that are tested to satisfy requirements.
- *Deploy*. The assembled business processes are deployed on the operating run-time environment.
- *Manage*. The services and the business processes that are executing on the run time are monitored and analyzed to ensure their smooth operations. That is to say, the entire service model is managed and monitored from IT and business perspectives. Information gathered during this phase is used to gain real-time insight into business processes, enabling better business decisions and feeding information back into the life cycle for continuous process improvement.

The entire process (see Fig.2) is controlled and orchestrated by the *governance* policies, to provide guidance and oversight for the target SOA application.

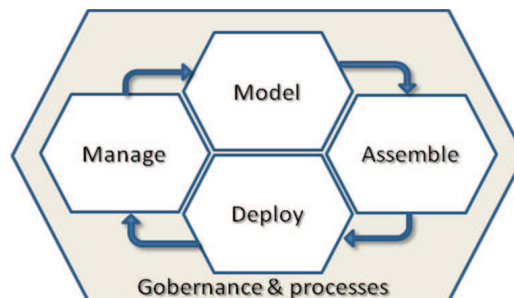


Fig. 2. SOA Lifecycle.

4 Services

Services are the building blocks of SOA enabled application. It is basically an encapsulation of data and business logic. A service consists of an interface, has an implementation and exhibits certain pre-defined behavior. The service interface defines a set of operations, which portrays (or exposes) its capabilities. Operations are the things that a service can do.

4.1 Classification

According to [Jus07] there are three common categories of services, and as a consequence it is easy to introduce different SOA layers and stages of expansion. Thus, services can be classified as:

- **Basic services** are services that provide basic business functionality, in such a way that it does not make sense to split them into multiple services. Usually, these services provide the first fundamental business layer for one specific backend or problem domain. The role of these services is to wrap a backend or problem domain so that consumers (and higher-level services) can access the backend by using the common SOA infrastructure.
- **Composed services** represent the first category of services that are composed of other services (basic and/or other composed services). In SOA terminology, composing new services out of existing services is called orchestration. They are typically services that access multiple backends and therefore are composed of multiple basic services. An example of a composed service would be a service that transfers money from one backend to another. In this case, the composed service would call one basic service that withdraws money from one backend and another basic service that pays the money into another backend.
- **Process Services** represent long-term workflows or business processes. A process service represents a long-running flow of activities (services) that is interruptible (by human intervention). Unlike basic and composed services, a process service usually has a state that remains stable over multiple calls. A typical example of a process service is a shopping-cart service: Its state would contain the contents of the shopping cart, perhaps combined with some customer data so that the customer's order could be maintained and manipulated over multiple sessions.

4.2 Service Identification

Services are at the core of SOA and we need to find what services are needed based on the requirements of clients. This process is called *service identification* [Ars04, Mitt06]. Services must be isolated and its scope fully delimited in terms of business impact.

Service identification has to deal not only with already existing services, but also with identifying services that do not yet exist. In consequence, identification can also result in the need to modify existing services, for example adding additional operations. Usually, it is difficult to identify all the services at once. It is better to proceed iteratively from a small initially identified set of services and gradually expand this set in subsequent iterations.

Generally, two approaches are applied to identifying services: *top-down* and *bottom-up*.

4.2.1 Top-down service identification

The top-down service identification approach (see Fig. 3) is business driven. You start identifying business processes that exists, and break them down based on functionality (i.e. functional areas or subsystems). In this step, business documentation can be helpful in the identification of services. Then, you have to determine which are the services needed and compare them with the services that exist in your organization. If they already exist, you simply can reuse, else you have to define the new services.

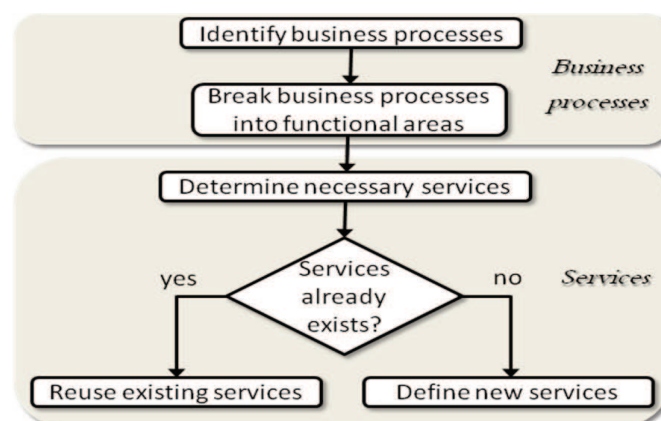


Fig. 3. Top-down service identification.

4.2.2 Bottom-up service identification

On the other hand, bottom-up service identification (see Fig. 4) is Information Technology (IT) driven. You start by digging deep into existing systems or applications looking for services (i.e. functionality is promoted as service). Each application is modeled as a set of services that are there because there is a concrete need for them. Then, the necessary services can be found in this set of services (i.e. you can reuse them) or not (i.e. you have to define them). In this approach difficulties arise due to problems with business documentation (strategy, processes, etc.), as it does not exist or is outdated.

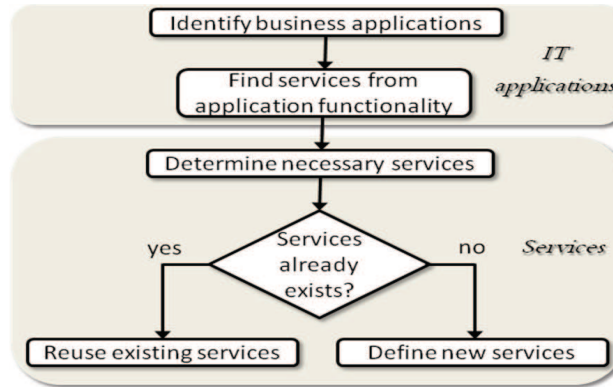


Fig. 4. Top-down service identification.

4.2.3 What approach to service identification should we use?

In reality, most organizations use a hybrid approach. It's important to understand that both approaches have different focus. The top-down approach is more business focused; whilst bottom-up is more IT focused. On one hand, services identified in a top-down approach have the risk of being too abstract to be useful. Whilst, on the other, services identified in a bottom-up approach have the risk of being too specific and might require some modifications to be useful.. So, the reason for the hybrid approach resides on the fact that SOA can succeed only if it is focused on both business and IT.

4.3 Service Analysis and Design

The issues included in the Open Group Service definition [OPE09] can be mapped against service design principles. Each one of them indicates the level of quality that a service needs to reach in order to be considered as a usable building block in the SOA we are trying to build. Table 1 shows the mapping between service definition and service design principles:

Table 1. Mapping of service definition issues against service design principles.

Service definition issue [OPE09]	Service design principle
<i>A logical representation of a repeatable business activity that has a specified outcome</i>	Provide value Meaningful Idempotent
<i>Self-contained</i>	Isolated
<i>Composed of other services</i>	Granularity Reusable Interoperable
<i>A "black box" to consumers</i>	Implementation hiding

Service design principles help service provider to create reusable services and help service consumer to judge if the services are well designed. The list below describes the mapped service design principles and can be used as a checklist when creating services [Dik12]:

- **Provide value** Consider if and why you need every service. If a service doesn't provide value to someone or something (clients, departments, other IT systems, and so on) then it is probably not a good service, or only part of a service and not a service in itself.
- **Meaningful** It should be easy for (future) consumers to use a service. Therefore the service interface needs to be meaningful to the *consumer and not too abstract or complex*. If a service is not meaningful, the required effort to consume a service will increase. Consumers will not be able or are reluctant to use such services since they don't understand them or it is too expensive to use and integrate them into their landscape.
- **Idempotent** A service should be predictable; invoking a service operation with the same input more than once should result in the same outcome.
- **Isolated** Services only provide flexibility and can only be easily changed if their operations are independent of other operations within the same or another service (this is *isolation* definition). If a change to an operation results in changes to several other operations that are tightly coupled to the originally changed operation, we lose flexibility. Operations need to be separate building blocks that provide capabilities themselves.
- **Granularity** Services are of different importance based on the degree of value or functionality they add.
- **Reusable** means a service can be used by more than one consumer.
- **Interoperable** Services should be easy to integrate into our IT landscape. Interoperability is a measure for the amount of effort it takes to use and invoke services. Interoperability is achieved by using standards for *describing, providing, and accessing* services such as XML, WS-*, WADL, and WSDL. Note that only the service interface needs to be interoperable; the implementation itself can be proprietary. For example, when you order breakfast in a coffee shop using consumer's language, you don't care what the cooks' language is, as long as they serve the breakfast you have ordered. Using standards to provide and access services helps to mix and match different technologies such as PL/SQL, .NET, Java, and PHP.
- **Implementation hiding** Consumers don't care about the actual implementation behind the service; this is a black box for them. Consumers focus on the contract and interface of a service to decide whether to use it and to be able to actually consume it. In short, a service and especially its interface and contract, should be self-describing and understandable.

Hiding implementation details is a common approach in several programming paradigms, even more so for SOA in which we specifically differentiate between contract, interface, and implementation. The service interface should abstract away (or hide) the specifics of the underlying systems and organizations that do the actual work. This makes it easier to change, upgrade, or swap the implementation without breaking interoperability since the interface can stay the same. It also doesn't burden consumers who don't need to know about the specifics of the implementation.

5 Get the Requirements for a Single Service: Service Specification

Service specification consists on capturing the requirements of a service. From a business point of view, the types of information (i.e. *business requirements*) that has to be gathered for specifying a service can be divided into the following categories [Mitt06]:

- **Accessibility** How the service can be found and accessed? You can start thinking about what are the processes that need to find and invoke the service you're building.
- **Functionality** What is the process or function this service provides? What business problem are you trying to solve? The appropriate granularity of the service has to be determined with respect to the degree of functionality added by the service to the SOA at hand.
- **Interaction** How does the service or application that calls this service interact with the service? How does the service handle error conditions?
- **Information** What data is sent to this service and back from this service?

- **Process** How it works? What are the relationships between the actions and events of this service?

Once it is know the information needed for a service specification, let get into the process. In SOA, the service specification process starts from the service providers (i.e. the stakeholders for whom you are creating a service). They have to describe what the service has to accomplish. That is to say, service providers describe service functionality using the information types described before and you document the requirements following a methodology.

In order to document requirements [Gra08, MAD14] for a single service we can use *use-cases* [Mitt06, Beh04, MAD14] *activity diagrams* and *BPMN* [Gra08]. The documentation process helps you validate the requirements with the stakeholders to get an agreement, and it helps the technical team that will implement the service. Figure 5 shows examples of use-case templates you can use to document these requirements for SOA.

SOA Use Case		Version:
Use case specification: <UC name>		Date:
Overview		
Use Case Name		
Description		
Started		
Pre-conditions		
Successful Post Conditions		
Trigger		
Actors		
Extensions		
Security		
Revision History		
Date	Version	Description
		Revised by
Assumptions		
Description	Origin	Date Created
Related Documents		
The following documents are supplementary to the use case and should be used in conjunction with the specification and implementation:		
Prototype:		
Requirements Doc.:		
Basic Flow -		
Description -		
Step	Actor Action	System Action
1		
2		
3		
Sub Variations - Alternatives		
Sub Variation Title -		
Description -		
Basic Flow Step N -		
SVA Step	Actor Action	System Action
1		
2		

a) [Mitt06] template

CU-016 Registrar préstamo																											
Versión	1.0 [02/07/2009]																										
Dependencias	<ul style="list-style-type: none"> RG-001 Gestionar los préstamos de los libros RG-005 Conocer las preferencias de los usuarios de la biblioteca RN- 008 Número máximo de préstamos simultáneos RN- 010 Fecha de devolución de un préstamo 																										
Precondición	El usuario de la biblioteca se ha identificado mediante su carné de biblioteca y ha cogido los libros objeto del préstamo de las estanterías.																										
Descripción	El sistema deberá comportarse como se describe en el siguiente caso de uso cuando el usuario de la biblioteca solicite al bibliotecario sacar uno o más libros en préstamo.																										
Secuencia normal	<table border="1"> <thead> <tr> <th>Paso</th> <th>Acción</th> </tr> </thead> <tbody> <tr> <td>1</td> <td>El bibliotecario solicita al sistema comenzar el proceso de registrar el préstamo de un libro.</td> </tr> <tr> <td>2</td> <td>El sistema solicita que se identifique al usuario de la biblioteca que desea retirar el libro.</td> </tr> <tr> <td>3</td> <td>El bibliotecario proporciona al sistema los datos identificativos del usuario de la biblioteca.</td> </tr> <tr> <td>4</td> <td>El sistema solicita que se identifiquen los libros objeto del préstamo.</td> </tr> <tr> <td>5</td> <td>El bibliotecario proporciona al sistema los datos de identificación de los libros objeto del préstamo.</td> </tr> <tr> <td>6</td> <td>El sistema muestra la fecha de devolución de cada uno de los libros objeto del préstamo y pide que se confirme el préstamo de cada uno de ellos.</td> </tr> <tr> <td>7</td> <td>El bibliotecario le indica al usuario de la biblioteca la fecha de devolución de cada libro y le pregunta si desea seguir adelante con el préstamo de los libros.</td> </tr> <tr> <td>8</td> <td>El usuario de la biblioteca confirma los libros que desea llevarse conociendo las fechas de devolución</td> </tr> <tr> <td>9</td> <td>Si alguno de los libros que se lleva tiene asociado un elemento multimedia, <table border="1"> <tr> <td>9.1</td> <td>Se realiza el caso de uso Añadir elemento multimedia al préstamo.</td> </tr> </table> </td> </tr> <tr> <td>10</td> <td>El bibliotecario confirma al sistema el préstamo de los libros que el usuario de la biblioteca ha decidido tomar prestado.</td> </tr> <tr> <td>11</td> <td>El sistema informa de que el préstamo de los libros se ha registrado correctamente.</td> </tr> </tbody> </table>	Paso	Acción	1	El bibliotecario solicita al sistema comenzar el proceso de registrar el préstamo de un libro.	2	El sistema solicita que se identifique al usuario de la biblioteca que desea retirar el libro.	3	El bibliotecario proporciona al sistema los datos identificativos del usuario de la biblioteca.	4	El sistema solicita que se identifiquen los libros objeto del préstamo.	5	El bibliotecario proporciona al sistema los datos de identificación de los libros objeto del préstamo.	6	El sistema muestra la fecha de devolución de cada uno de los libros objeto del préstamo y pide que se confirme el préstamo de cada uno de ellos.	7	El bibliotecario le indica al usuario de la biblioteca la fecha de devolución de cada libro y le pregunta si desea seguir adelante con el préstamo de los libros.	8	El usuario de la biblioteca confirma los libros que desea llevarse conociendo las fechas de devolución	9	Si alguno de los libros que se lleva tiene asociado un elemento multimedia, <table border="1"> <tr> <td>9.1</td> <td>Se realiza el caso de uso Añadir elemento multimedia al préstamo.</td> </tr> </table>	9.1	Se realiza el caso de uso Añadir elemento multimedia al préstamo.	10	El bibliotecario confirma al sistema el préstamo de los libros que el usuario de la biblioteca ha decidido tomar prestado.	11	El sistema informa de que el préstamo de los libros se ha registrado correctamente.
Paso	Acción																										
1	El bibliotecario solicita al sistema comenzar el proceso de registrar el préstamo de un libro.																										
2	El sistema solicita que se identifique al usuario de la biblioteca que desea retirar el libro.																										
3	El bibliotecario proporciona al sistema los datos identificativos del usuario de la biblioteca.																										
4	El sistema solicita que se identifiquen los libros objeto del préstamo.																										
5	El bibliotecario proporciona al sistema los datos de identificación de los libros objeto del préstamo.																										
6	El sistema muestra la fecha de devolución de cada uno de los libros objeto del préstamo y pide que se confirme el préstamo de cada uno de ellos.																										
7	El bibliotecario le indica al usuario de la biblioteca la fecha de devolución de cada libro y le pregunta si desea seguir adelante con el préstamo de los libros.																										
8	El usuario de la biblioteca confirma los libros que desea llevarse conociendo las fechas de devolución																										
9	Si alguno de los libros que se lleva tiene asociado un elemento multimedia, <table border="1"> <tr> <td>9.1</td> <td>Se realiza el caso de uso Añadir elemento multimedia al préstamo.</td> </tr> </table>	9.1	Se realiza el caso de uso Añadir elemento multimedia al préstamo.																								
9.1	Se realiza el caso de uso Añadir elemento multimedia al préstamo.																										
10	El bibliotecario confirma al sistema el préstamo de los libros que el usuario de la biblioteca ha decidido tomar prestado.																										
11	El sistema informa de que el préstamo de los libros se ha registrado correctamente.																										
Postcondición	El usuario de la biblioteca se lleva los libros prestados y el sistema ha registrado el préstamo de los libros.																										
Excepciones	<table border="1"> <thead> <tr> <th>Paso</th> <th>Acción</th> </tr> </thead> <tbody> <tr> <td>3</td> <td>Si el usuario de la biblioteca ha excedido el número máximo de préstamos simultáneos o tiene alguna penalización, <table border="1"> <tr> <td>E.1</td> <td>El sistema informa de la situación que impide realizar el préstamo.</td> </tr> <tr> <td>E.2</td> <td>El bibliotecario retiene los libros e informa al usuario de la biblioteca de la situación.</td> </tr> <tr> <td>E.2</td> <td>Se cancela el caso de uso.</td> </tr> </table> </td> </tr> </tbody> </table>	Paso	Acción	3	Si el usuario de la biblioteca ha excedido el número máximo de préstamos simultáneos o tiene alguna penalización, <table border="1"> <tr> <td>E.1</td> <td>El sistema informa de la situación que impide realizar el préstamo.</td> </tr> <tr> <td>E.2</td> <td>El bibliotecario retiene los libros e informa al usuario de la biblioteca de la situación.</td> </tr> <tr> <td>E.2</td> <td>Se cancela el caso de uso.</td> </tr> </table>	E.1	El sistema informa de la situación que impide realizar el préstamo.	E.2	El bibliotecario retiene los libros e informa al usuario de la biblioteca de la situación.	E.2	Se cancela el caso de uso.																
Paso	Acción																										
3	Si el usuario de la biblioteca ha excedido el número máximo de préstamos simultáneos o tiene alguna penalización, <table border="1"> <tr> <td>E.1</td> <td>El sistema informa de la situación que impide realizar el préstamo.</td> </tr> <tr> <td>E.2</td> <td>El bibliotecario retiene los libros e informa al usuario de la biblioteca de la situación.</td> </tr> <tr> <td>E.2</td> <td>Se cancela el caso de uso.</td> </tr> </table>	E.1	El sistema informa de la situación que impide realizar el préstamo.	E.2	El bibliotecario retiene los libros e informa al usuario de la biblioteca de la situación.	E.2	Se cancela el caso de uso.																				
E.1	El sistema informa de la situación que impide realizar el préstamo.																										
E.2	El bibliotecario retiene los libros e informa al usuario de la biblioteca de la situación.																										
E.2	Se cancela el caso de uso.																										
Comentarios	El número máximo de préstamos simultáneos y la duración de los préstamos depende de la política de la biblioteca y puede cambiar en el futuro. Ver las reglas de negocio RN-008 y RN-010.																										

b) [MAD14] template

Fig. 5. Use case templates for service oriented requirements.

Diagramas de secuencia?? K Mittal 2006; T Behrens 2004

6 Methodology for service oriented requirement engineering

Defining software requirements is recognized as critical for a software project's success. An important factor in software development failure is insufficient or erroneous requirements management [Gla02, Ema08]. The computer-programming community defines requirements activity as engineering in itself. Therefore, requirements engineering can be defined as the set of processes required for reaching an agreement between developers, customers and users regarding the intended functionality of a planned system; together with the acceptance criteria definition that allows stakeholders to decide whether the complete system is valid or not.

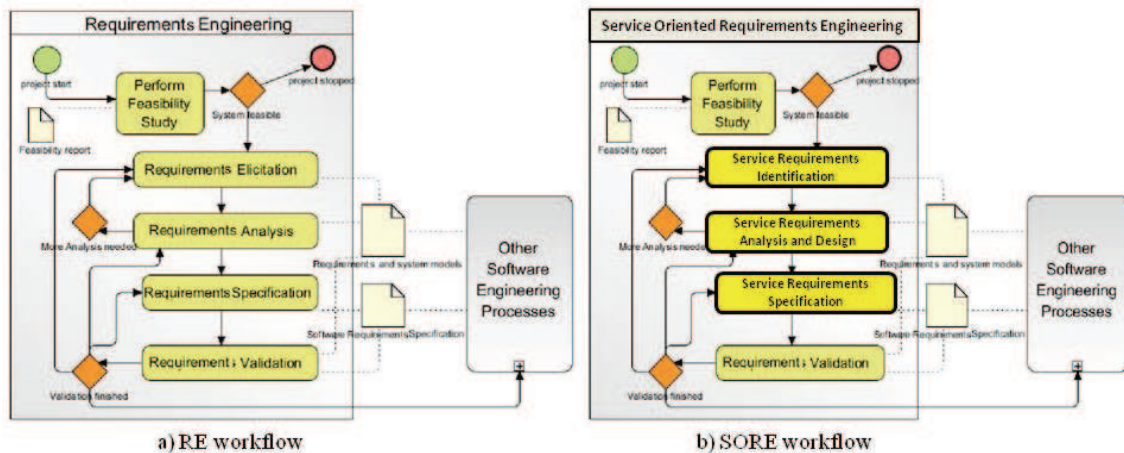


Fig. 6. Requirements Engineering and Service Oriented Requirements Engineering workflows.

The requirements stage consists of several activities that have to be carried out in a software development project. These activities can be organized as a workflow (see Figure 6a). This workflow unifies the main methodological approaches. It starts with a feasibility study that is constructed in a way that determines the project scope and the available resources. After this, software developers execute an elicitation cycle, and then analyze, specify and validate software requirements until ending up with a valid software requirements specification. This document subsequently serves as the baseline for the software development steps [IEEE98]. The previous workflow together with the processes associated to services (i.e. *identification, analysis and design, and specification*) can be adapted to deal with service oriented requirements defining a service oriented requirements engineering workflow (see Figure 6b).

Requirements are elicited from users through interviews and other techniques such as questionnaires or brainstorming. This often proves a complex task because activities requiring human communication usually imply problems in understanding; whereas the concept of requirements is to define without ambiguity what the system is expected to do. Consequently, developers are usually tasked with analyzing and refining requirements in order to obtain a valid baseline.

The requirements captured are gathered in a document or its electronic equivalent, known as Software Requirements Specification (SRS) [IEEE98]. This tedious and prone to error task is performed within the requirements specification stage, usually with the aid of a software tool for the management and maintenance of a large set of requirements specifications.

Requirements validation is performed to check if the elicited and specified requirements present any inconsistencies, whether the information is complete or not, and whether there are any ambiguities in the system definition. This process (elicitation-analysis-specification-validation) is carried out over several iterations until deciding if the requirements specification has been successfully completed.

References

- [OAS06] OASIS SOA Reference Model TC, [OASIS Reference Model for Service Oriented Architecture 1.0](http://docs.oasis-open.org/soa-rm/v1.0/soa-rm.pdf), Official OASIS Standard (Normative PDF), Oct. 12, (2006) <http://docs.oasis-open.org/soa-rm/v1.0/soa-rm.pdf>
- [OPE09] The Open Group SOA Working Group, [Service Oriented Architecture : What Is SOA?](http://www.opengroup.org/soa/source-book/soa/soa.htm) January (2009) 4th Ed., <http://www.opengroup.org/soa/source-book/soa/soa.htm>
- [Abu08] Khalil A. Abuosba, Asim A. El-Sheikh, "Formalizing Service-Oriented Architectures", *IT Professional*, vol.10, no. 4, July/August, (2008) 34-38, doi:10.1109/MITP.2008.70
- [Jia09] Aihua Jia; Wenfeng Li; Dingfang Chen; Xiaowei Zhang, "Research of SOA-Based dynamic enterprise workflow integration platform," *1st IEEE Symposium on Web Society, SWS '09*, 23-24 Aug, (2009) 184-188, doi:10.1109/SWS.2009.5271787
- [Jus07] Nicolai M. Josuttis, *SOA in Practice: The Art of Distributed System Design*, O'Reilly, Sebastopol, CA (2007) ISBN-10: 0-596-52955-4
- [Mitr06] T. Mitra. *Architecture in practice, Part 1: Realizing Service-Oriented Architecture*, IBM developersWorks (2006) <http://www.ibm.com/developerworks/webservices/library/ar-arprac1/index.html>
- [Dik12] Lonke Dikmans, Ronald van Luttikhuisen, *SOA Made Simple: Discover the true meaning behind the buzzword that is 'Service Oriented Architecture'*, Packt Publishing, Birmingham, UK (2012) ISBN-10: 1849684162
- [Hig05] Rob High, Jr., Stephen Kinder, Steve Graham, *IBM's SOA Foundation: An Architectural Introduction and Overview, Version 1.0*, November (2005)

- [Gra08] I Graham. Requirements modeling and specification for service oriented architecture, John Wiley & Sons Ltd, Chichester, UK (2008)
- [Mitt06] K. Mittal, Requirements process for SOA projects, Part 2: Business requirements for your first SOA services, IBM developersWorks (2006) <https://www.ibm.com/developerworks/library/ar-soareq2/>
- [Ars04] A. Arsanjani, Service-oriented modeling and architecture. How to identify, specify, and realize services for your SOA, IBM developersWorks (2004) <http://www.ibm.com/developerworks/library/ws-soa-design1/>
- [Gla02] RL Glass, Software Engineering: Facts and Fallacies. Addison-Wesley Longman Publishing Co., Inc., Boston, MA, USA (2002)
- [IEEE98] I IEEE.: IEEE Recommended Practice for Software Requirements Specifications. Tech. rep. (1998) URL http://ieeexplore.ieee.org/xpls/abs_all.jsp?arnumber=720574
- [Ema08] K El Emam, A Koru, A replicated survey of it software project failures. Software, IEEE 25(5), 84-90 (2008)
- [MAD14] Junta de Andalucía, MADEJA: Marco de Desarrollo de la Junta de Andalucía, Versión 1.5.0 (2014) URL <http://www.juntadeandalucia.es/servicios/madeja/>