

A Method for Bridging the Gap between Business Process Models and Services

Leonardo Guerreiro Azevedo^{1,2}, Flávia Santoro¹, Fernanda Baião¹, Thaíssa Diirr¹, Alexandre Souza¹, Jairo Francisco de Souza³, Henrique Prado Sousa⁴

¹ Graduate Program in Informatics (PPGI), Department of Applied Informatics
Federal University of the State of Rio de Janeiro (UNIRIO).
458 Pasteur Av., Rio de Janeiro, RJ, 22290-240, Brazil

² IBM Research
146 & 138 Pasteur Av., Rio de Janeiro, RJ, 22290-240, Brazil

³ Department of Computer Science
Federal University of Juiz de Fora (UFJF), Brazil

⁴ Pontifical Catholic University of Rio de Janeiro, Brazil

{azevedo, flavia.santoro, fernanda.baiao, thaissa.medeiros,
alexandre.souza, henrique.souza}@uniriotec.br,
jairo.souza@ufjf.edu.br, lga@br.ibm.com, hsousa@inf.puc-rio.br

Abstract. *Many proposals in the literature are consensual in making business processes as the starting point of a Service-Oriented system development life-cycle. However, there is no systematic approach that can be easily applied in practice. We argue that an effective SOA approach requires an integrated view of organizational business processes, where services are explicitly related to business models components. Accomplishing these requirements is vital for bridging the gap between business needs and their supporting services. This work proposes a top-down method for service identification and analysis from business process models. Each step of the method implements a set of heuristics that are also specified. The method is presented in detail, and constitutes a systematic guide for service identification and analysis. A case study is conducted to demonstrate the use of the method in practice.*

1. Introduction

SOA (Service-Oriented Architecture) is a paradigm for the development and maintenance of business processes that span large distributed systems (Josuttis 2007). Service is the core concept in SOA. Papazoglou *et al.* (2007) define services as autonomous, platform-independent entities that can be described, published, discovered, and loosely coupled in novel ways. A service may perform functions that range from simple requests to sophisticated business process. Any piece of code and any application component deployed on a system can be reused and transformed into a network-available service.

Service orientation promises to significantly improve the manageability and changeability of increasingly complex information systems (Aier *et al.* 2011). In addition, SOA has been advertised as an answer to enable a flexible alignment of

business needs and IT capabilities (Trkman *et al.* 2011). Despite its enormous technical promises, the deployment of SOA in an organization poses a series of challenges, such as the introduction of new architectural roles and development tasks and the need for specifying a service development approach that explicitly takes organizational business needs and their corresponding process models into account (ARIS Paper 2007, Arsanjani 2004, Erl 2005, Fareghzadeh 2008, Gu and Lago 2007, Josuttis 2007, Klose *et al.* 2007, Kohlborn *et al.* 2009, Scheer 2000).

Within the service development life-cycle, service identification and analysis are the first and the most important phases to foster business/IT alignment. Many works address the derivation of services from fully-automated business processes; however, real processes typically mix system-intensive and human-intensive activities, which should also be considered for service derivation. Moreover, the same activity may appear in several processes throughout the organization, supported by several information systems in different areas or departments. Hence, it raises the need for an integrated view of organizational business processes, where each process is related to the organizational key-value chain. In this integrated view, the relationships among processes, and among elements within the same process, are explicit.

This paper proposes a top-down method for service identification and analysis, which are the first and the most important phases of the service development life-cycle in fostering business/IT alignment. The method works on top of a common repository of inter-related business process models following the EPC notation, and is composed by a set of systematic steps that are presented in a rich detail level to enable its application in practice. The method returns a list of the proper services to be further designed and implemented. An important advantage of this phase is that it follows a top-down approach by making explicit links from the elements of business process models to intermediary results, and then to the final candidate services. Those explicit links are useful for traceability purposes, for example to track changes and quantify the impacts produced by a new business requirement. The resulting set of services is inherently aligned to the set of organizational business processes, thus reducing the Business/IT gap. The proposed method was evaluated in a case study to exemplify the application of each step and provide a better understanding of the whole approach.

The rest of the paper is organized as follows: Section 2 discusses related work (life-cycle approaches); Section 3 presents a proposal for Service Life Cycle; Section 4 describes the details of a case study used to evaluate the proposal; and, Section 6 concludes the paper and points future research perspectives.

2. Related Work

Typical activities in software development process are: requirement elicitation, requirement analysis, design, implementation, testing and deployment. Roles that usually perform those activities are project managers, analysts, designers, software architects, developers, customers and quality assurance. A life-cycle model corresponds to the definition of a specific thread of activities for software development. Examples of traditional software development life-cycle models are cascade, iterative, incremental, component-based, spiral, and Rapid Application Development (Pressman 2006).

Business process models may support the development of an information system and present approaches to derive requirements from such models (De la Vara González and Sánchez Díaz 2007). However, conventional software engineering life-cycle models are not directly applicable to SOA. New roles and architectural development tasks and new challenges are introduced due to service-oriented development characteristics (Gu and Lago 2007). For example, a life-cycle model should consider service provider, service consumer and service broker architectural roles. Examples of new challenges are: how to deal with conflicting requirements; how to align business requirements to IT solutions; how to distribute services across organizational boundaries in a secure manner. Thus, an specific life-cycle model is vital for proper SOA implementation (Pulier and Taylor 2006).

Gu and Lago (2007) and Kohlborn *et al.* (2009) evaluated a large number of service life-cycle supporting methodologies proposed by different research approaches from both academia and industry. They concluded that there is no consensus on how life cycle should be conducted. Proposals vary widely in scope and details. Kohlborn *et al.* (2009) state that the evaluated proposals do not cover business services and software services in a comprehensible and integrated manner.

Gu and Lago (2007) analyzed some models (McBride 2007, Papazoglou and Heuvel 2006, Sun 2006, Systinet 2006, Tsai *et al.* 2007, Wall 2006), and proposed a service development life-cycle as a sequence of steps grouped into three phases: *design time*, *run time* and *change time*. The design phase occurs before a service is made available for use. During the implementation phase, services are available to run. The change phase deals with new requirements and errors found after the implementation stage. In their approach, service identification is not handled explicitly, even though it is essential for service modeling.

Kohlborn *et al.* (2009) propose a method to support service life-cycle. Their approach is structured as follows: (1) Derivation of business services (Preparation Phase; Identification Phase; Detail Phase; Prioritization Phase); (ii) Derivation of software services (Preparation Phase; Identification Phase; Identify corresponding entities; Analyze and visibility takeover; Identify potential service operations; Extract process logic; Define logical contexts; Define service compositions). The method is a good guideline for organizations and practitioners, including dealing with various issues relevant to service identification. However, they do not provide enough details to conduct this step.

Inaganti and Behara (2007) suggest the identification as the first step of the life-cycle of service-oriented development. Service identification is often a challenging activity for application development teams, because there is no business process documentation, and expertise is necessary to identify service characteristics. These authors warn mistakes in identification may lead to errors in design and implementation activities. Consequently, multiple iterations may be required, especially in composing services to be used by applications.

Arsanjani (2004) proposes business process modeling using SOMA method - a top-down strategy. SOA is more strategic and business-aligned, while web services are a strategic implementation. Although modeling steps are presented in his approach,

activities are not described in detail and there is no systematic approach for identifying services from business processes models.

Jamshidi *et al.* (2008) address aspects of initial steps in building service-based solutions – mainly service modeling, considering business models models. This approach assumes the business process model is highly detailed (up to the level of Elementary Business Process - EBP) and the granularity of each business entity is the same as the EBP which creates it. Process models at such a level of abstraction are not easy to accomplish. Also, this proposal does not consider other process model elements (e.g., business rules, business requirements, process flows).

Adam *et al.* (2008) propose a method for deriving web services systematically based on the business processes of a representative sample of organization's partners. Service identification takes place at an appropriate level of abstraction at which a precise set of functions is stipulated. "Set of functions" refers to the number of different operations provided by the service, while the "level of abstraction" describes whether a transaction executes a more business-oriented feature or a more technical functionality.

Birkmeier *et al.* (2013) propose a method for web services identification based on business process models. Eventhough they present a systematic approach, the initial set of functionalities identified from the business process to be considered in the services definition is highly dependent on the analysts definition. The business analyst identifies which activities of the business process should be automated. In addition, s/he groups those process activities which belong together in a logical perspective. As a result, an initial list of functionalities representing the before-mentioned activity groups is provided. The list summarizes all functionalities that should be examined for implementation. To generate this list, business analysts use existing business process models to identify the functionalities that are relevant in the specific project context; however, this relevance criteria are not detailed in their proposal.

Bianchini *et al.* (2013) proposed a well detailed and systematic method for web services identification based on business process models. The proposed method is very similar to the method proposed in our previous works (Azevedo *et al.*, 2009a) (Azevedo *et al.*, 2011), which is being improved and revised in this work. Bianchini *et al.* (2013) uses CRUD (Create, Retrieve, Update, Delete) operations and workflow patterns, but they do not consider important information present on process models that is useful to identify candidate services, such as business rules and business requirement.

We conclude the related work is highly dependent on the expertise of the SOA analyst to find a suitable set of services meeting specific business requirement. There are very few systematic methods for service identification during business analysis. The proposals presenting a step in this direction pose some limitations, since they are not detailed enough, do not provide specific guidelines for their execution or leave out important aspects of business process models.

Our proposal complements the approach of Gu and Lago (2007) by providing a method to address two crucial steps of the service development life-cycle: service identification and analysis.

3. A Proposal for Service Identification and Analysis based on Business Process Models

We present a method based on business process models designed using EPC notation (Keller and Teufel, 1998; Scheer, 2000) to support service identification and analysis in software organizations that implement SOA. Our recommendation is to derive services from business process models considering their particular characteristics in accordance to SOA principles. The method makes use of several important concepts defined below.

We classify services into candidate services *versus* physical services. A physical service is a functionality that is implemented in a programming language, while a candidate service is an abstract (not implemented) service which, during the design phase of a service life-cycle, can be chosen to be implemented as a service or as an traditional application function (Erl 2005).

We define two types of candidate services, namely candidate data service and candidate logic service. A candidate data service only performs CRUD (Create, Retrieve, Update and Delete) operations on data. A candidate logic service implements a behavioral business rule, such as a mathematical formula or a condition test. Business rule defines or restricts some aspect of an organization (BRG 2001). It aims to establish the structure of a business, control or influence its behavior. Structural business rules are responsible for defining domain concepts and constrain how these concepts may be related to each other, thus regulating how organization systems should handle data sources (Cserie *et al.* 2009). Behavioral business rules restricts business behavior by limiting actions that may take place in a specific scenario, or stating formulae and inference rules to derive knowledge from existing concepts.

The proposed method is divided into service identification and service analysis steps. Both steps explicitly take information from business process models into account, which fosters Business-IT alignment. The present work considerably improves preliminary and partial versions of each step, which were respectively proposed by Azevedo *et al.* (2009a) and Azevedo *et al.* (2011), by adding new heuristics and revising previous ones.

For each step, we precisely specify a set of heuristics to be executed systematically for service identification and analysis. The proposed service identification heuristics cover all workflow patterns specified by Russel *et al.* (2004) and Van der Aalst *et al.* (2003). Service analysis heuristics follow principles of high-quality service implementation (ARIS Paper 2007, Arsanjani 2004, Erl 2005, Jamshidi *et al.* 2008, Josuttis 2007, Klose *et al.* 2007, Marks and Bell 2006). The result from those steps is a list of services at the most appropriate granularity level, defining the degree of reuse and specifying development prioritization aligned to business needs. In addition, groups of services are defined. They are divided into data service groups and logical service groups.

Service Identification Phase generates a list of candidate services and information about them. This information serves as input for Service Analysis Phase, in which the candidate services are prioritized, grouped and designed. These two phases are presented in Section 3.1 and Section 3.2. Section 4 presents a case study that exemplifies the use of the heuristics and demonstrates the effectiveness of the approach.

Heuristics were developed using EPC notation (Keller and Teufel, 1998; Scheer, 2000) for business process modeling. This notation is presented in Appendix 1. However, the majority of heuristics can be directly applied on business process models designed using other notation, such as BPMN (Ko *et al.* 2009), which was developed and standardized by the Object Management Group (OMG 2011).

3.1. Service Identification Phase

The first phase of our proposed service life-cycle is the identification of candidate services from a set of business process models. The main steps are: (i) Selection of Activities; (ii) Identification and classification of candidate services; (iii) Consolidation of candidate services .

Service identification starts when a demand for software development is received. Demands comprise set of requirements to be implemented (either as services or application functions). Other inputs for the method are: “to-be” business process models (re-designed processes in which the new software requirements are proposed and represented), and a set of business requirements already implemented in existing applications. Figure 1 presents the input and output information of service identification phase. To-Be business process models are used for identification and classification of candidate services, while business requirements of existing systems and business requirements of the demand are used for candidate services consolidation. The outcome of the method is a list of candidate services along with a set of elements (tables, charts, service dependency graphs) that should assist the Service Analyst in making decisions as to the most suitable implementation for each identified candidate service. The list of candidate services serves as input for the next steps in a service life-cycle model (analysis and design).

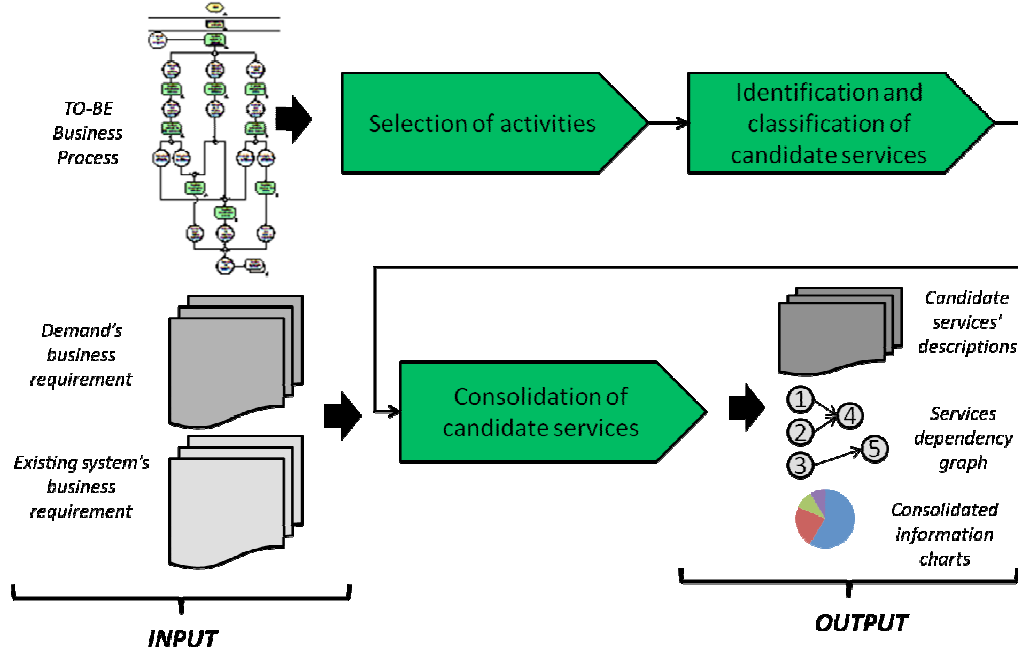


Figure 1. Input and Output information of Service Identification Phase

“Selection of activities” step (Figure 2) elects a set of business activities of To-Be business process models for possible service support. The classification of Le Clair and Teubner (2007) was used. It divides all business processes into either human-intensive or system-intensive. Human-intensive processes require people to get work done while relying on and interacting extensively with business applications, databases, documents and other people via collaboration tools. They require human intuition or judgment for decision-making during individual steps in the business process. System-intensive processes typically involve a large number of transactions with minimal or no human intervention.

All activities of system-intensive processes are selected. Activities of a human-intensive process not considered for automation are discarded, while those which may be system-supported are selected.

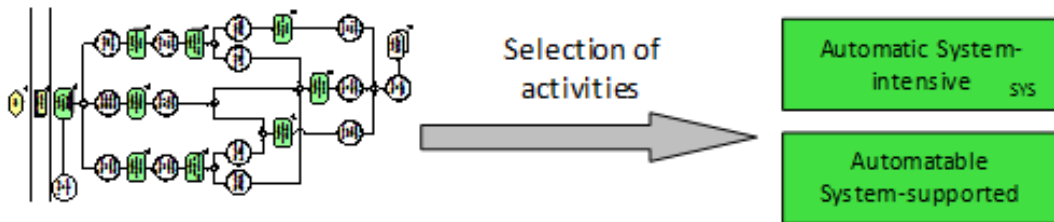


Figure 2. Selection of Activities step

In the second step (Figure 3), candidate services are identified by applying a set of heuristics to the set of activities selected in step 1. The proposed heuristics were defined to address both syntactic and semantic analysis of the business process model.

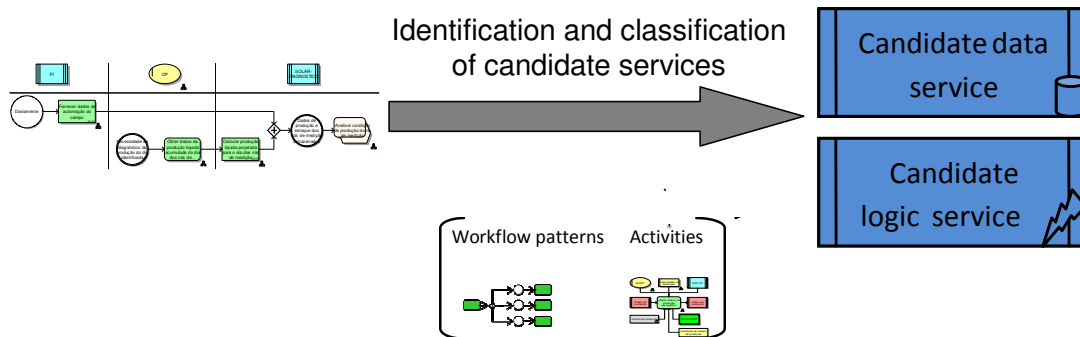


Figure 3. Identification and Classification step

Semantic analysis of business process models should consider the intended meaning of each element represented in the process model. Some elements denote useful information for providing a (service-based) computational support for the process. Considering all possible business process elements (Sharp and McDermott 2001), three semantic element types are addressed: (i) “activity input and output data”; (ii) “business requirement”; and (iii) “business rule”. Three service identification heuristics are proposed (Heuristic 1 to 3). Each heuristic must be applied to business

process models in which the corresponding element type appears related to one of the process activities selected in the “Selection of activities” step.

Syntactic analysis of business process models is carried out by considering the process model structure, i.e., from each workflow pattern specification proposed by Russel *et al.* (2004) and Van der Aalst *et al.* (2003) (Heuristics 4 to 10). The goal is to assure coverage of the flows that may be represented within a process model.

- *Semantic analysis heuristics*
 - **Heuristic 1 (Business Rule):** A candidate service must be identified from a business rule.
 - **Heuristic 2 (Business Requirement):** A candidate service must be identified from a business requirement.
 - **Heuristic 3 (Input/Output Information):** A candidate service must be identified from input and output information that are linked to information support element.

An exemplary application of semantic analysis heuristics is presented in Figure 4. This model corresponds to a Function Allocation Diagram of activity “Verify client data”. The activity has three business rules: “Outdated client data”; “New client”; “Client identification”. Three candidate services are identified from for these elements using heuristic H1. Two candidate services are identified from the business requirements “Consult client’s information” and “Consult credit proposal” using heuristic H2. Finally, two candidate service are identified from “Client register” and “Credit proposal” input information using heuristic H3.

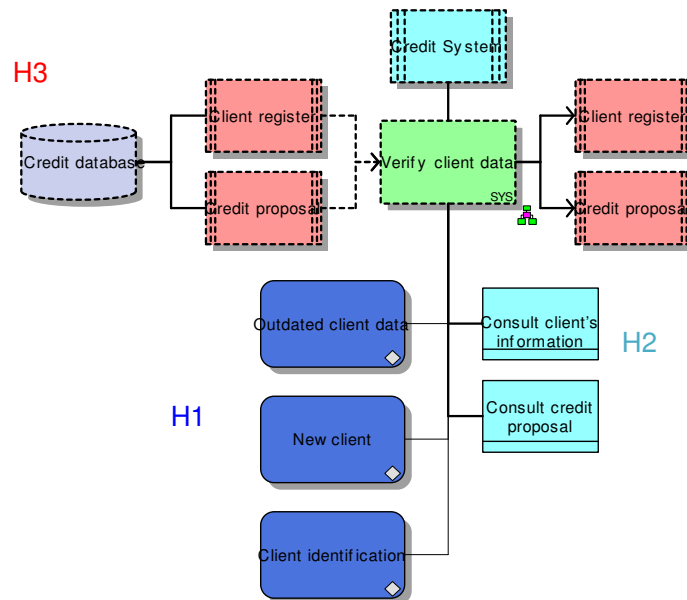


Figure 4. FAD model of “Verify client data” activity.

- *Syntactic analysis heuristics*

- **Heuristic 4 (Sequence of Activities):** A candidate service must be identified from a series of sequential activities.
- **Heuristic 5 (AND):** A candidate service must be identified from an AND-pattern.
- **Heuristic 6 (XOR):** A candidate service must be identified from a XOR-pattern
- **Heuristic 7 (OR):** A candidate service must be identified from an OR-pattern.
- **Heuristic 8 (Loop):** A candidate service must be identified from a Loop-pattern.
- **Heuristic 9 (Process Interface):** Candidate services must be identified from the automated interaction between two processes: one candidate service to pass the information to the other process, and another candidate service to receive that information.
- **Heuristic 10 (Multi-Instance Activity):** Candidate services must be identified from a multi-instance activity: one candidate service to send the information to each instance of the multi-instance activity; one candidate service to perform each instance of the multi-instance activity; and one candidate service to consolidate the outputs of the instances and to pass the result to the next step.

The syntactic analysis heuristics identify candidate services based on workflow patterns. The candidate service can encompass more than one activity from the business process. Each logic candidate service encompasses the activity that starts the logical operator (AND, XOR, OR, loop) and all the activities inside the control block. Samples of candidate service identification from workflow patterns are presented in **Figure 5**. Candidate services (1), (2) and (3), identified by Heuristics 5, 6, and 7, respectively.

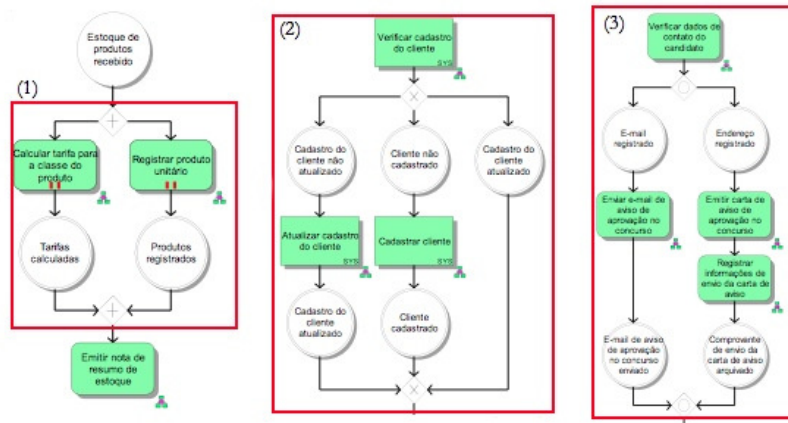


Figure 5. Examples of part of business process models where candidate services were identified considering control flow heuristics.

Candidate services are identified by conducting both semantic and syntactic analyses (that is, by applying Heuristics 1 to 10) on top of business processes models selected previously. Heuristics may be applied in any order, since the information processed by each heuristic is independent. However, for automation purposes, we indicate handling together heuristics that identifies candidate services from activity elements, which are concerned to semantic analysis. In other words, we suggest implementing a loop that applies all semantic heuristics for each activity. Syntactic analysis heuristics may be applied in any order.

Once identified, each candidate service is classified as a candidate data service or a candidate logic service, according to the previous definition. Each candidate service identified by the heuristics is described by the following attributes:

- **Identifier:** a sequential unique number that may be generated automatically.
- **Name:** candidate service name, e.g., “Verify if customer data corresponds to a new registration”.
- **Type:** “candidate logic service” or “candidate data service”, e.g., “Verify if customer data corresponds to a new registration” could be classified as data candidate service.
- **Heuristics:** name of the heuristics that discovered the candidate service, e.g., “Business Rule Heuristic”.
- **Input:** input information required for service execution, e.g., “customer identifier”.
- **Output:** output information generated by the service, e.g., “indication that customer is already registered”.
- **Activities:** name of all business activities that are somehow related to this service. There are two possible scenarios to define this set of activities: (a) when the candidate service was identified by a semantic analysis heuristic (in this case, there is an element that originated the candidate service), this set is composed by all activities to which this element is related to; (b) when the candidate service was identified by a syntactic analysis heuristic (in this case, a set of activities instantiated the workflow pattern that originated the candidate service), this set is composed by all activities that are part of this workflow pattern. For example, if the candidate service “Verify if customer data corresponds to a new registration” was identified from a business rule associated to two activities in the process repository (“Register client” and “Verify client data”), then its “Activities” includes both. If a candidate service “Generate credit proposal” was identified from a sequence of activities “Compromise credit limit”; “Calculate tax rate”; “Determine interest rate to be charged”; “Generate contract proposal”; “Analyze contract”, they are all included into its “Activities” set.
- **Description:** a description in high level of what the service must execute. E.g., “Verify if customer data corresponds to a new registration” should query customer information database to verify if customer already exists.

Figure 6 exemplifies an algorithm for Heuristic 4, which identifies a candidate service from a sequence of activities. A sequence of activities is composed by two or more activities that are executed sequentially, that is, with no logical operators (OR, XOR or AND) among them (van der Aalst *et al.*, 2003). The algorithm in Figure 6 invokes a method that returns all sequences of activities in a business process model. For each sequence, if all its activities are supported or executed by a system, then a candidate service is created and described using the attributes identifier, name, type, heuristic name, input/output parameters, list of activities of the sequence.

```

identifyCandidateServiceSequentialAct(processModel)
  seqActs receive all set of sequential activities existing in
  processModel
  for each seqAct in seqActs
    if all activities of seqAct are supported or executed by
    a system then
      create candidate service from seqAct

```

Figure 6. Automation of Heuristic 4

In the third step, candidate services information is consolidated by applying a set of service consolidation heuristics (Heuristics 11 to 16). Service consolidation aims at gathering information for each candidate service about pre-defined criteria that helps a Service Analyst to decide upon its implementation. Services that are not selected for implementation are removed, resulting in a refined list of candidate services.

The proposed service consolidation heuristics were based on high-quality service implementation principles widely-known in the literature (ARIS Paper 2007, Arsanjani 2004, Erl 2005, Jamshidi *et al.* 2008, Josuttis 2007, Klose *et al.* 2007, Marks and Bell 2006). Thus, they reflect the most important technical issues that should be considered by analysts. In addition, information regarding candidate service usage (by process activities or other candidate services) and existing implementations is also considered.

- *Service consolidation heuristics:*
 - **Heuristic 11 (Service Reuse):** The candidate service reuse is calculated as the sum of times the service is used by each process activity.
 - **Heuristic 12 (Link Candidate Service and System):** A candidate service that is identified from a business requirement already implemented must be linked to the systems that implement the requirement.
 - **Heuristic 13 (Link Candidate Service and Demand Requirements):** A candidate service identified from a business requirement of the client demand must be linked to the requirement.
 - **Heuristic 14 (Link Candidate Service and Activities):** A candidate service must be associated with the activities from which it was identified.
 - **Heuristic 15 (Identify Candidate Services Dependencies):** A candidate service must be associated with other candidate services that use it.

- **Heuristic 16 (Identify Candidate Utility Service):** A candidate utility service must be identified from observation of recurrent patterns (Thom *et al.* 2007).

Association among business requirements and systems (Heuristic 12) allows the analyst to identify candidate services identified from developed requirements and to prioritize the service development according system's demand priorities. As a business requirement can be implemented by more than one system, it is not ease to identify the association among business requirements and systems visually. Thus, the association is identified by Heuristic 12 from the system business requirement model.

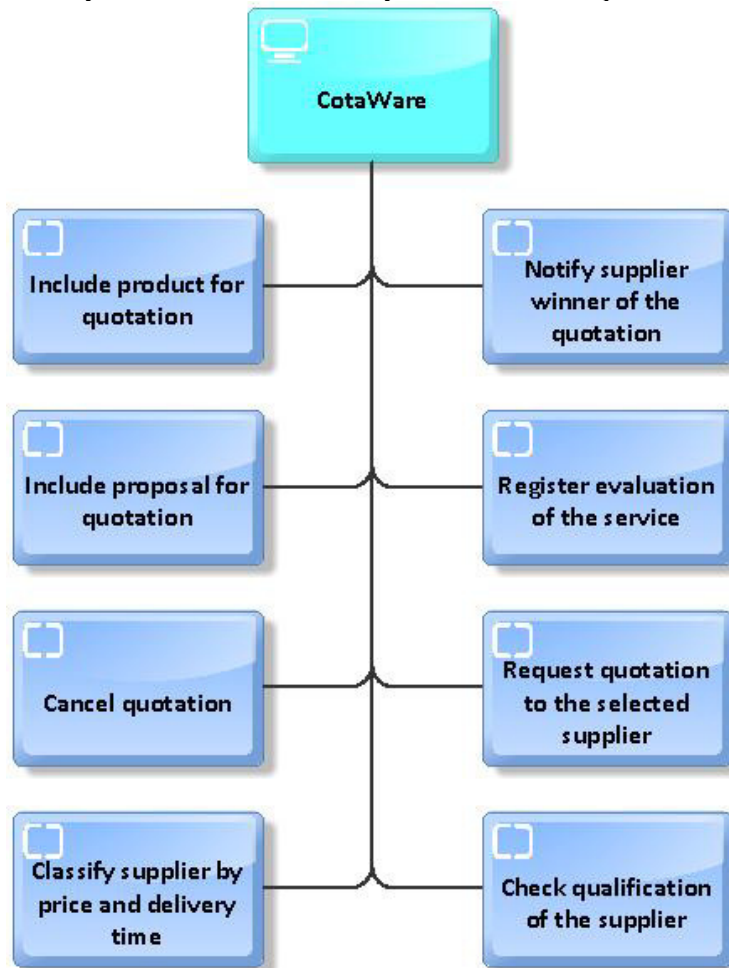


Figure 7 present an example of business requirement model which is used to link candidate services identified from the business requirements to CotaWare system.

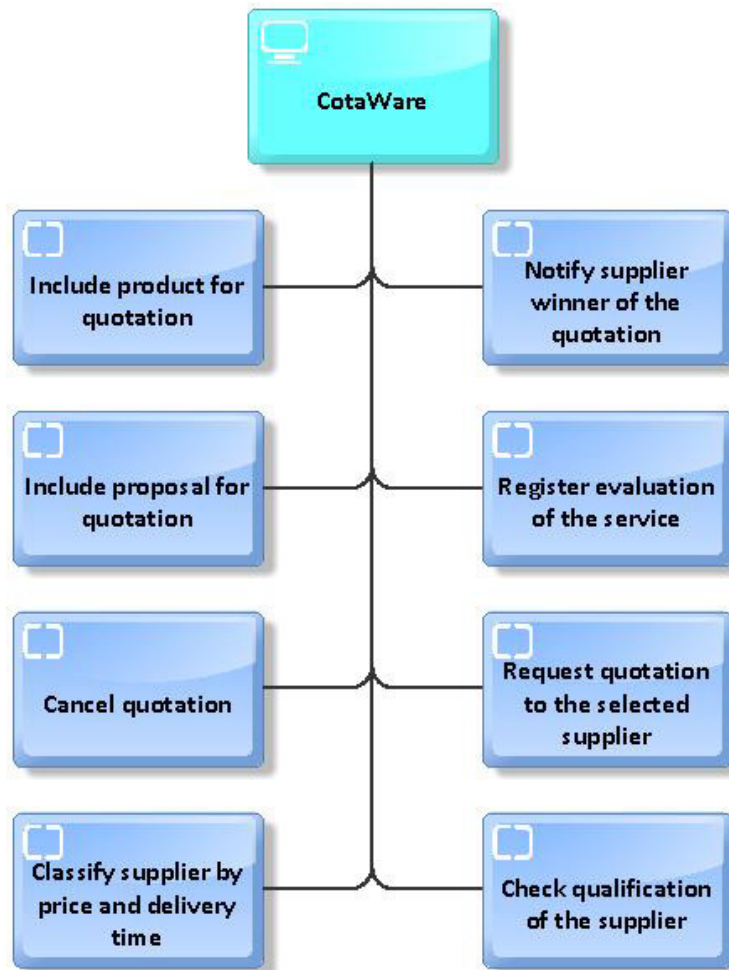


Figure 7. Business requirement model from CotaWare system

The association among candidate services and demand requirements (Heuristic 13) allows the analyst to identify the set of services that are necessary to meet the initial demand. Moreover, the association among candidate services and process activities (Heuristic 14) eases the understanding of the service context. In this case, it is possible to identify candidate services that encompass more than one process activity. Services identified from FAD model presented in Figure 4 would be related to the activity “Verify client data”.

The final result from the identification phase is the refined list of candidate services with their corresponding attributes, as well as tables, charts, and dependency graphs including reuse information for service analysts. The information gathered by these heuristics is summarized in a table, such as, for example Table 1.

Table 1. Candidate services information

id	Candidate Service Name	Reuse	Multiple Instance	Roles	Systems	Process activities	Demand Req.
2	Verify if client data corresponds to a new registration	2	No	Role1	Apl1	Register client Verify client data	Yes
25	Execute tax rate calculus	1	No	Role5	Apl1	Calculate tax rate	Yes
35	Treat credits granted	5	No	Role2 Role1	Apl1	Approve contract Compromise credit limit Cancel contract Cancel risk contract	Yes

3.2. Service Analysis Phase

This section details the service analysis. Input is a set of candidate services resulting from service identification phase described in Section 3.1. Output is a set of services at the most appropriate granularity. They will be designed for implementation (the next phase of the method). The service analysis phase consists of the following steps: (i) Prioritize candidate services; (ii) Define granularity of candidate services; (iii) Group candidate services.

3.2.1. Prioritize candidate services

The goal of candidate service prioritization is to identify which candidate services may contribute most to the organization and, therefore, should initially be considered for analysis. Candidate service prioritization is achieved by automatically analyzing the components of the business process models, thus reducing the possible subjectivity of this process. Just as the identification phase is based on business process models, this prioritization phase prioritizes services according to business characteristics, i.e., which system is more important or can achieve greater ROI for the company or which services decrease redundancy. Therefore, according to one heuristic one candidate service could have a high score, and according to another one it could have a minor value. At the end, the order reflects the best choices. The heuristics are presented below, and are illustrated in the exploratory case study. Without automatic prioritization, SOA analyst must examine hundreds of services to determine which it is most important to implement. We try to support the analyst by providing heuristics that scan the BPM repository and calculate priority based on weights that can be set by the analyst, as follows.

- **Heuristic 17 (Candidate service reuse level):** Group candidate services by level of reuse: list reuse level and number of services at each reuse level, and then define grades for each level. Afterwards, assign a weight to each candidate service.

The higher is the candidate service reuse level, the greater is the gain from implementing the service, since it reduces code replication and keeps functionalities in a central place. For example, the implementation of a candidate service with reuse level of 1 as a physical service does not add much gain to the organization (note, however, that in specific scenarios in which this precise service is used only in one process but that process is critical for the organization, or if it presents a high maintenance cost, the SOA designer may further decide for its implementation). Therefore, the weight for this service would be zero. However, services with a reuse level of 8 will probably bring major gain to the organization. Thus, for instance, on a scale of weights from 1 to 5, one could assign weight 5 to services at reuse level 7 and 8. An example of weights associated by analysts for each reuse level is presented in Table 2.

Table 2. Example of weights by reuse level

reuse level	# candidate services	weight
1	77	0
2	9	1
3	17	3
4	1	3
7	2	5
8	1	5

- **Heuristic 18 (Association of candidate services with systems):** Group each candidate service linking to the system that implements it: list systems and the number of candidate services related to each one, and then define grades for each system. Afterwards, assign weights to each candidate service.

In this case, setting the weights depends more on the importance of the system to the organization than on the number of services associated with the system. In addition, during the identification phase, the candidate service should be marked as implemented.

- **Heuristic 19 (Association of candidate services with systems they rely on):** Group candidate services that could potentially be invoked by systems: list systems and number of candidate services related to each system, and then define grades for each system. Afterwards, assign weights to each candidate service.

Following the reuse principle, it is important to identify the potential use of candidate services by systems, as well those existing systems in the business processes studied, and others that were not considered before. In this particular case, the analysis should also include services that are no longer used by any system, since systems could be changed to use them. Furthermore, it is also important to make a broader analysis, including other potential systems that may use the services. An example of weights associated by analysts for each system is presented in Table 3.

Table 3. Example of weights by system.

system	# candidate services	weight
Apl1	14	2
Apl2	84	3
Apl1, Apl2	1	5
(none)	11	1

- **Heuristic 20 (Increase candidate service weights according to multiple instantiation activities):** Increase the weight of candidate services identified from multiple instantiation activities.

Services identified from multiple instantiation activities tend to be used much more than others, since they are reused by each instance of the activity. Therefore, these services should have greater weight. For instance, the priority of services identified from multiple instances activities could be increased by 1.

- **Heuristic 21 (Association of candidate services with demand):** Increase the weight of candidate services that are associated with the client demand requirements.

Candidate services identified from business process models are not necessarily directly related to the client demand requirements, because demand may bear on only a part of the process model. Thus, different weights should be assigned to each of these cases.

- **Heuristic 22 (Association of candidate services with roles):** Group candidate services that are associated with each business role: list business roles and number of candidate services related to each one, and then define grades for each role. Afterwards, assign weights to each service.

Associating candidate services with business roles highlights the importance of the service in terms of the importance of the roles. Candidate services performed by systems should have greater weight, because implementation of the feature will be fully automated, without human intervention or need for GUI development. Services that require human intervention entail a user interface. These services should have lower weight. Table 4 contains an example of business role group weights.

Table 4. Example of business role groups

Roles	Business Unit	# candidate services	weight
Apl1; Role1; Role2	UN1; UN3	1	2
Apl2		50	5
Role1; Apl2	UN1; UN4; UN3; UN2	22	2
Apl1		13	5
Role2; Role1; Apl2	UN3; UN1; UN2;	1	3

After defining the weights, service priority is calculated by consolidating produced values as stated by Heuristic 23. The sum of priorities helps to indicate which are the most important services to the business, regarding several perspectives. For example, after summing the weights, a service with a reuse level of 1 may have higher priority than a service with a reuse level of 5. This may happen if the service relates to a very critical application, is executed in a multiple instance activity and/or is executed by important roles in the organization.

- **Heuristic 23 (Calculation of candidate service prioritization):** Sum the weights produced by each prioritization heuristic (Heuristics 17 to 22) to define service priorities.

An example of service prioritization is presented in Table 5. The last column is calculated using Heuristic 23.

Table 5. Example of candidate service prioritization

Service	Weights						Priority (H23)
	Multiple Instance (H20)	Reuse (H17)	System Implementation (H18)	System Reuse (H19)	Demand (H21)	Roles (H22)	
#74	0	0	2	4	1	5	11
#41	0	1	3	4	1	5	14
#95	0	2	1	4	1	2	10
#30	0	5	2	8	1	2	18
#92	0	5	5	4	1	3	18

An important concept in SOA is service composition. Papazoglou *et al.* (2007) presents that service composition aggregates multiple services into a single composite service. Resulting composite services can be used as basic services in further service compositions or offered as complete applications and solutions to service consumers. Besides, Papazoglou *et al.* (2007) emphasize that “orchestration” and “choreography” are the most used terms to describe business interaction protocols that coordinate and control collaborating services. Orchestration describes how services interact at the message level, including the business logic and execution order of interactions under control of a single end point. It is an executable business process that can result in a long-lived, transactional, multistep process model. Choreography is typically associated with message exchanges, rules of interaction, and agreements that occur between multiple business process end points rather than a specific business process executed by a single party.

We propose heuristics for service composition identification from business process workflows (Heuristics 4 to 8), and we propose the following heuristic to prioritize this sort of candidate service.

- **Heuristic 24 (workflow candidate service prioritization):** Prioritize workflow services by: number of activities that compose the workflow; number of models where the same workflow is reused; number of entities handled by the workflow; number of different lanes; and number of sub-workflows. For each case, weight should be assigned according to a priority range to demonstrate priority. Afterwards, sum the weights produced by each prioritization in order to produce a consolidated value.

For example, knowing that the number of activities that make up the flow ranges from 2 to 25, we define the following weights: (i) 0 for 0-2 activities; (ii) 1 for 3-5 activities; 2 for 6-10 activities; 3 for 11-15 activities; 4 for 16-20 activities; and 5 for 21-25 activities. An example of workflow candidate service prioritization is presented in Table 6.

Table 6. Example of workflow candidate service prioritization

Service	Weights					Priority (H24)
	# automated activities	# supported activities	# models	# entities	#sub-workflows	
#110	4	0	1	-	-	5
#120	0	2	1	-	1	4

3.2.2. Service granularity

After prioritizing the candidate services, the next step is to analyze their granularity. We propose to develop a granularity map similar to the one proposed by Marks and Bell (2006), in which services are located on a map according to their granularity. The dependency graph produced in the service identification step (Heuristic 15) is used to place services on this map. The dependency information was derived from the relationships among the business processes elements from which the candidate services were identified. For example, business requirements often refer to business rules. So, a dependency relationship is created between candidate services identified from business rules (which are finer-grained) and candidate services identified from business requirements (which are coarser-grained). Another example is that the (coarser-grained)

services identified from flows are directly related to the (finer-grained) candidate services identified from the elements in each activity. These dependencies can be identified automatically, if there are explicit relationships among the process model elements, which is a premise of this work. The heuristic for candidate service granularity is:

- **Heuristic 25 (Candidate service granularity):** Develop a granularity map for candidate services and place services on that map so that coarser-grained services are at the top and finer-grained services are at the bottom.

An example of granularity map for candidate services is presented in Figure 8. The reuse value is calculated by Heuristic 17.

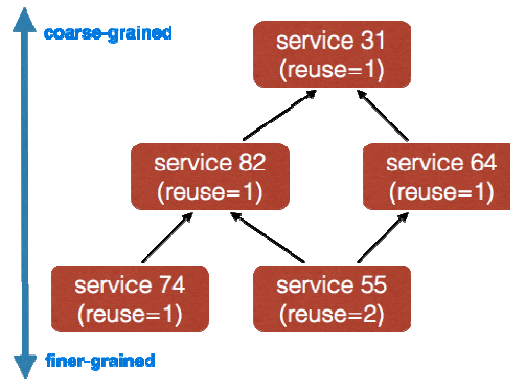


Figure 8. Candidate service analysis step

We propose a mechanism to help the SOA analyst to get a good, yet preliminary idea of granularity. Using the granularity map together with information about service importance, potential service reuse, current service use, among other relevant information, the SOA analyst can decide what service granularity is most appropriate in the analysis phase. In the design phase, information about hardware, bandwidth, processing etc. is introduced, and granularity can change. For example, coarse-grained services can be decomposed into smaller ones, while finer-grained services can be grouped or included in other services in order to suit granularity to physical constraints. However, even though the granularity defined during the analysis phase is not final, we argue that it is very close to that. As a result, costs incurred by design changes resulting from implementation issues and maintenance are reduced, since better analysis is performed.

3.2.3. Defining service groups

Data services (also called entity services) are very generic and reusable, since they encapsulate data access operations (CRUD - Create, Read, Update or Delete) that can be performed by any application. Fareghzadeh (2008) points out that these sorts of services are not tightly coupled to business processes, given that they provide an interface that is more oriented to data than to procedures. Thus, these services must be analyzed in terms of the entities they manipulate. These entities can be identified directly from the business process models: for example, Azevedo *et al.* (2011) propose listing the entities associated with data models relating to activity input and output information. If this information is not available, then it is necessary to examine existing entity-relationship

or class diagrams, or to interview data administrators and users in order to elicit the entities manipulated by services.

The candidate services are listed in a table, and then linked to the entities (and corresponding attributes) they manipulate. Afterwards, candidate services that execute operations on the same entity or group of entities are generalized into a single candidate service. The heuristic for grouping data services is:

- **Heuristic 26 (Grouping Data Services):** Associate candidate data services with entities, indicating the CRUD operations that the service performs on these entities. Then, group services which executes at least one of Create, Update and Delete operation over an entity group. For each group of services that execute only read operation on an entity group, define a communication channel between the service group and the service group that manipulates the entity group. Design services in such a way as to reduce the communication cost between them.

Groups of logic services must take into account the use context of the logic embedded in the service.

- **Heuristic 27 (Logic service groups):** Group logic services by standards in the organization.

A proposal for step-by-step implementation of this heuristic is presented below:

1. Identify the most abstract concept used by the organization in different areas.
2. Identify candidate logic services dealing with the context of this concept (and related concepts), considering the views of different areas of the organization.
3. Generalize candidate services that perform very similar operations on the concept.
4. Group candidate logic services that operate on the concept context into the same package (or the same service).

An example of logic service groups is presented in Table 7. The values C, R, U, D are CRUD (Create-Retrive-Update-Delete) operations. The value H is set when a service group accesses an entity handled by another service group. In this case, the services should access the entity through invocation of the services that encapsulate the entity.

Table 7. Example of service grouping

Service		Entity groups						
		Entity 1	Entity 2	Entity 3	Entity 4	Entity 5	Entity 6	Entity 7
G1	Service 14	C				H		
	Service 1	R						
	Service 3	R						
G2	Service 18		C					H
	Service 2		C					
G3	Service 7		H		C			
	Service 6				R			
	Service 13				R			
G4	Service 9		H			C		
	Service 8					R		
	Service 12					R		
	Service 17					R		
G5	Service 10		H				C	
	Service 11		H				C	
G6	Service 4			C				
	Service 5			C				
G7	Service 16		H					R
	Service 15							R

The data type models used by the service should be developed to prevent replication of the same data types for different projects, to increase reuse and to document data types.

If the organization develops a canonical model, it is important that data type models used by the service reference the canonical model, indicating which classes and attributes are used. A canonical information model is a model of the semantics and structure of information that adheres to a set of rules agreed upon within a defined context for communicating among a set of applications or parties (Gilpin, 2010). Reference between models can be made through explicit dependencies between classes and attributes or by markings on the canonical model. Figure 11 presents an example of data model.

- **Heuristic 28 (Data type model used by the service):** Model the data types used by the service in a UML diagram. If a canonical model is used, explain the relationship between classes and attributes of data types and classes of the canonical model.

4. Application Scenario

This section presents an application scenario as a pilot evaluation of the proposal. According to Bishop *et al.* (2007), thinking over the future profoundly and creatively is critical, or instead, we assume the risk of being surprised, other than not being ready to face changes. Likewise, given the uncertainty and inability to predict everything that can happen, we have to run through several plausible settings, not only what we expect to take place. Being so, a methodological approach to deal with this kind of issue is defined by Bishop *et al.* (2007) as the scenario building: “*scenario is the archetypical product of futures studies because it embodies the central principles of the discipline*”. Based on a broad literature review, the authors also states a scenario contains “*the stories of these multiple futures, from the expected to the wildcard, in ways analytically coherent and imaginatively engaging*”.

The business process model “Analyze credit request”, presented in Appendix 2, was used as the application scenario and comprises 18 activities carried out by 2

departments in a financial organization. The goal of this business process is to analyze credit proposals, which as a result are approved or denied. When a credit proposal is received, the client information is checked and the system verifies whether the client credit limit is compatible with granting the proposed credit. If the limit is OK, then the system calculates fees and taxes to generate a contract proposal. This contract proposal is routed to an analyst who identifies any need for adjustment, and the level of risk entailed by the loan. If the contract is acceptable, it is sent to an approval group that will contact the client to assess it. Once the contract is approved it will be ratified. If the contract is not approved, then it is canceled. The full process is presented in more details by Diirr *et al.* (2010). Here, we present a small subset of the workflow, in order to follow the application of the method.

Besides the flow of activities, it's also important knowing the following information about each activity (since the proposed method requires them): input and output information, business rules, business requirements, information supporting elements, roles that perform the activity and systems that support the activity. Figure 9 shows an example of diagram for the “Generate contract proposal” activity where the representation of such information is highlighted with dashed lines.

These process and activity models were designed using the EPC notation presented by Keller and Teufel (1998) and Scheer (2000).

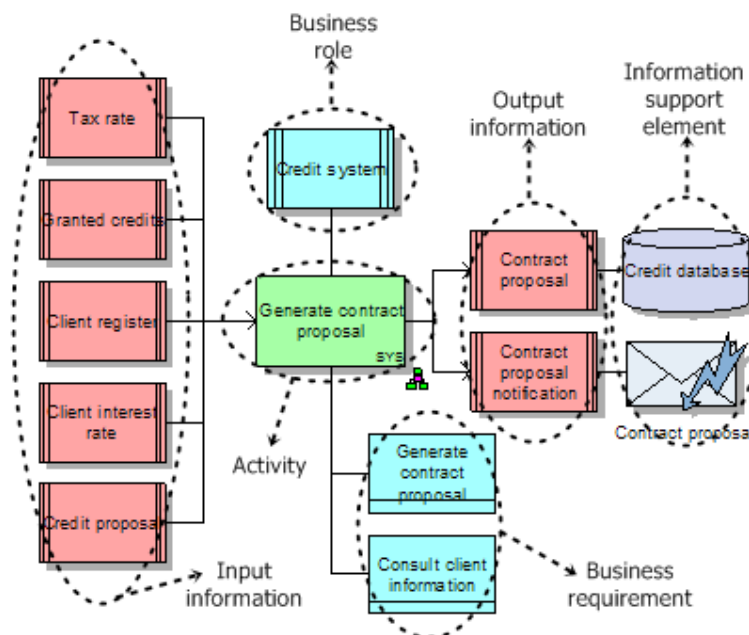


Figure 9. Model of the “Generate contract proposal” activity

4.1. Defining service groups

When the Service Identification Phase heuristics were executed on the “Analyze Credit Request” business process model, 47 services were identified and described. A partial list of candidate services is presented in Table 8. These services were identified using Heuristics 1 to 10. Information in the third to sixth columns of Table 8 results from the application of Heuristics 11 to 15. This partial list includes services identified from

input/output information, business rules, multiple instance activities, XOR pattern etc. In order to demonstrate an example of heuristics use, we present how service #32 and service #36 were identified.

Service #32 was obtained, according to Heuristic 4, from a sequence of activities presented in Figure 9. This sequence occurs only once and consists of the following activities: “Compromise credit limit”, “Calculate tax rate”, “Determine interest rate to be charged”, “Generate contract proposal” and “Analyze contract”. This information is in “Reuse” and “Activities” columns, respectively. The activities that compose the sequence flow are supported by Credit system and it is executed by Credit System and Credit Analyst roles. Besides, this service was not identified from a multiple instance activity and comprises demand requirements. This information is, respectively, in “System”, “Roles”, “Multiple instance” and “Demand Requirements” columns.

Service #36 was identified using input and output information from activity models, according Heuristic 3. The information “Contract proposal” occurs nine times linked to an information support element in the following activities: “Analyze contract”, “Approve contract”, “Cancel contract”, “Cancel risk contract”, “Check contract conditions with client”, “Generate contract proposal” (Figure 9) and “Notify proposal not approved”. This information is in “Reuse” and “Activities” columns, respectively. These activities mentioned above are supported by Credit system and executed by Attendant and Credit System roles. Besides, none of these activities is a multiple instance activity and the service identified comprises demand requirements. This information is, respectively, in “System”, “Roles”, “Multiple instance” and “Demand Requirements” columns.

Table 8. Candidate services identified

Candidate Service		Reuse	Multiple Instance	Roles	Systems	Activities	Demand Req.
2	Verify if client data corresponds to a new registration	2	No	Credit System	Credit System	Register client	Yes
						Verify client data	
25	Execute tax rate calculus	1	No	Credit System	Credit System	Calculate tax rate	Yes
28	Calculate tax rate	1	Yes	Credit System	Credit System	Calculate tax rate	Yes
32	Generate credit proposal	1	No	Credit System	Credit System	Compromise credit limit	Yes
				Credit Analyst		Calculate tax rate	
						Determine interest rate to be charged	
						Generate contract proposal	
						Analyze contract	
35	Treat credits granted	5	No	Attendant	Credit System	Approve contract	Yes
				Credit System		Compromise credit limit	
						Cancel contract	
						Cancel risk contract	
36	Treat contract proposal	9	No	Attendant	Credit System	Notify proposal not approved	Yes
				Client		Generate contract proposal	
						Check contract conditions with client	
						Analyze contract	
						Approve contract	
						Cancel contract	
Credit System	Cancel risk contract						
38	Treat credit proposal	3	No	Credit System	Credit System	Cancel credit proposal;	Yes
				Credit Analyst		Modify credit proposal;	
						Verify client data;	

4.2. Service Analysis Phase

Using the information resulting from the Service Identification Phase, weights were calculated for services in order to prioritize the services on a scale from 0 to 5. The top 5 priority services are shown in Table 9. The weights considered service reuse (Heuristic 17 – column Reuse), relationship between systems and services (Heuristic 18 – column System), if the service was identified from multiple-instance activity (Heuristic 20 – column Multiple Instance), and relationship between business roles and services (Heuristic 22 – column Roles). Weights were assigned considering the following rules which were defined by SOA analysts:

- Services with reuse equals to 1 received weight 0; reuse equals to 2 or 3 received weight 1; reuse equals to 5 received weight 3; and reuse equals to 9 received weight 5.
- Services supported by Credit System received weight 1;
- Services identified from multiple instance activity received weight 2;
- Services identified from activities executed by:
 - Client and Attendant, or Attendant role received weight 1;
 - Credit Analyst and Attendant, or Credit Analyst: received weight 2;
 - Credit System and Attendant, or Credit System received weight 4;
 - Credit System and Credit Analyst received weight 5;
 - Credit System, Credit Analyst, Attendant and Client received weight 6.

The priority of each candidate service is given by summing all weights (Heuristic 23 – column Priority). The prioritization step resulted in a list of candidate services, ranked by priority in decreasing order. For example, service #36 (presented in Table 8) has reuse equals to 9 and was not identified from a multiple instance activity. So, it received weigh 5 in “Reuse” column and 0 in “Multiple instance” column. Besides, this service was identified from activities supported by the “Credit System” and executed by “Credit System”, “Credit Analyst”, “Attendant” and “Client” roles. Then, it received weigh 1 in “System” column and 6 in “Roles” column. Therefore, the priority of service #36 was calculated as 12.

Table 9. Top 5 priority services

Service ID	Candidate Service Name	Reuse	System	Multiple Instance	Roles	Priority
36	Treat contract proposal	5	1	0	6	12
27	Generate credit proposal with adjustment	0	1	2	5	8
35	Treat credits granted	3	1	0	4	8
45	Analyze credit request	0	1	0	6	7
47	Analyze credit request with approved proposal	0	1	0	6	7

The granularity map (Heuristic 25) for the “Generate credit proposal with adjustment” composite service is presented in Figure 10. In this diagram, arrows represent dependencies. The dependencies were discovered using service identification heuristics. This diagram shows:

- Composite services: service #27 invokes services #6, #23, #32 and #38, and service #32 invokes service #28. So, it is easy to see service dependencies using this map.
- Reuse: service reuse levels are highlighted by circles. For example, service #6 has reuse equals to 2.
- Sub-composites: shaded rectangular backgrounds indicate composite services, and sub-levels are represented by different colors. In this example, service #27 invokes service #32. Service #32 is a composite service, and it invokes service #28, which is another composite as well.

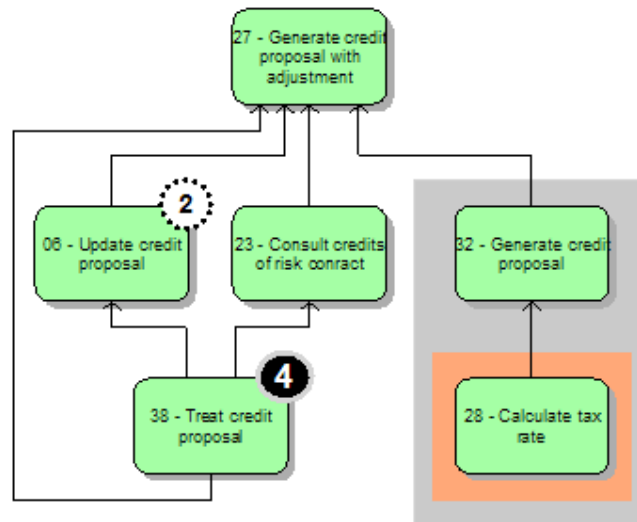


Figure 10. Granularity map of “Generate credit proposal with adjustment service”

Figure 11 shows a portion of the canonical model that was developed (Heuristic 28). The canonical data model is used to understand the concepts handled by services, and is used to group services according to the entities they manipulate (Table 10).

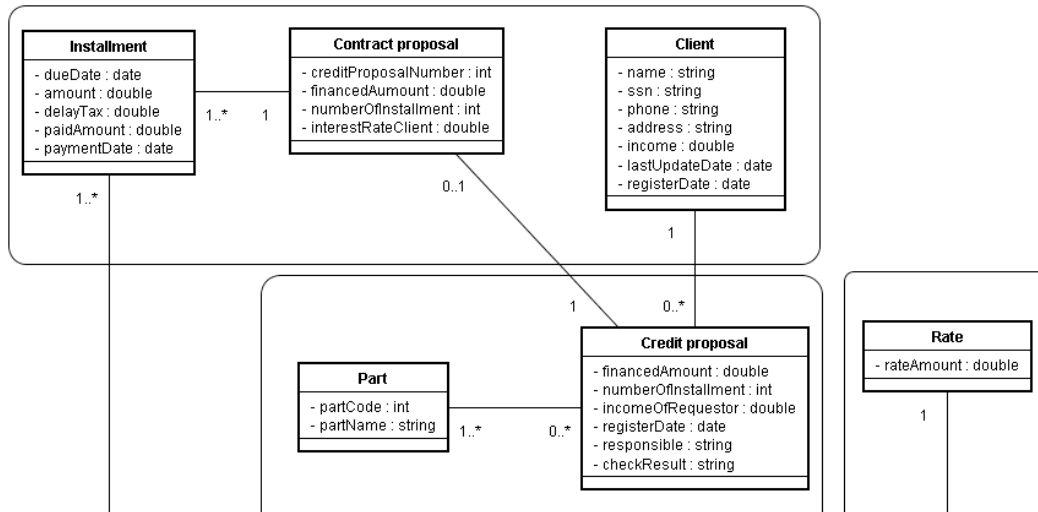


Figure 11. Part of the canonical data model

The CRUD table was built according to the operations that each data candidate service performs on each entity (Heuristic 26). A small subset of this table is presented in Table 10. This table is used to group services that execute mainly create, delete and update operations on the same entity, or group of entities. As services #36 and #43 operate on “Contract proposal” entity, both were grouped together. Using the same reasoning, services #37 and #44 were grouped together, since they operate on “Client” entity. All of this information is used to analyze candidate services, and results in a proposal of which services should be implemented. This proposal should be analyzed further considering other information, like performance issues. In this case, the proposal obtained in Table 10 was accepted, because services (#43) and (#37) have low reuse and will not impact in the performance of services (#36) and (#44) that have higher reuse.

Table 10. CRUD table

Services		Group of entities						
		Instalment	Contract proposal	Client	Credit proposal	Credits granted	Part	Rate
Group 1	36 Treat contract proposal	CR	CRU		-	-	-	-
	43 Treat client interest rate		C		-	-	-	-
Group 2	37 Treat client credit limit			R	-	-	-	-
	44 Manage client data			CRU	-	-	-	-

This work presents details about some candidate services that were identified and analyzed using our proposals. However, other operations were performed on the other candidate services, and they are presented as follows:

- Some candidate services identified from business rules or business requirements that only execute CRUD operations that are already performed by other candidate data services were excluded.
- Services that have high reuse were indicated to be implemented separately.
- Some services that have low reuse and only are invoked together were grouped into a coarser grained service.
- The composite service, identified according to workflow pattern, and responsible for managing client data was grouped together with other services

which have low reuse and represent functionality very related to this specific context.

- The service that executes all operations concerning contract proposal generation was indicated to be implemented separately from other services.
- Some very simple composite services, comprising two or three simple services, were identified according to workflow patterns. Since they are very simple compositions, they were not considered for implementation as physical services.

Finally, in this case study, application of the method resulted in 8 services to be implemented, selected from the 47 candidate services identified in the first step:

- #35 (Treat credits granted);
- #36 (Treat contract proposal);
- #38 (Treat credit proposal);
- #3 (Check credit limit);
- #15 (Generate contract proposal);
- #28 (Calculate tax rate);
- #42 (Treat tax rate);
- #44 (Manage client data);

5. Conclusions and Future Research Perspectives

The deployment of SOA in an organization poses a series of challenges, especially for identifying and specifying a set of services that adequately supports business needs. Proposals for service development life-cycle are typically too generic and fail to take an integrated view of organizational business processes into account. A service life-cycle not only facilitates management of service-oriented systems, but also improves service governance.

Among the activities of a service life-cycle model, we emphasize the service identification and analysis steps. This work proposes a top-down method from EPC business process models that implements a set of heuristics to derive services from business process definitions. The method considers both semantic and syntactic perspectives of process models, in order to bridge the gap between IT support and business activities. Semantic analysis takes business rules, data elements and business requirements into account, while syntactic analysis derives services from business process models according to their structural patterns. The result is a set of candidate services that are identified, scored, sorted, and grouped according to a set of service quality criteria (reuse, existing implementations, part of the demand requirements for development, use by process activities, relationship between services, and service granularity). This helps service analysts to better design and implement services, while also addresses subjective issues such as security, policy decisions, and so on. Moreover, explicit links are defined from business process elements and service descriptions. This improves traceability, and allows for precisely estimating impacts or the required effort

when a business process evolve, and also trigger changes in the services that support the process, and vice-versa. The association of services and business process models helps in tracing any changes within the models that directly impact IT, as well as changes in IT that should be reflected in the business process models. Furthermore, changes in processes can trigger automatic changes in service implementation.

The proposed method was assessed in a case study in which a business process model for credit request analysis was used as input. The service identification step was applied manually, and resulted in identification of 47 candidate services. Service analysis resulted in 8 services to be designed.

Preliminary versions of the service identification heuristics were applied in a real scenario for Oil Production Diagnosis (Azevedo *et al.*, 2009a; Azevedo *et al.*, 2011). The services identified were validated by system analysts who developed the physical services to support process automation. All the services they implemented were identified by our proposal, and specialists agreed that the consolidated information helped them in implementing the services.

As future work, we intend to apply the proposed method in large-scale case studies to demonstrate its scalability in different domains. We are also studying the subsequent steps (design, implementation, deployment and maintenance) and changes in business process models resulting from service life-cycle steps. Other important future work relates to the service reuse principle. A service can be reused in a variety of contexts; in other words, it can support different business processes. So, a change in a business process can entail service maintenance, and consequently impact other business processes. Management of these changes and business process impacts raises important issues. The descriptions provided in the present work may also be used to specify a supporting tool to automate its execution. In fact, a preliminary prototype that partially supports the identification step was presented by Azevedo *et al.* (2009b). This tool was implemented using ARIS Report Script, a module of ARIS SOA Architect Platform (<http://www.ids-scheer.com>), which includes methods for querying elements from business process models in a repository.

References

- Adam, S., Riegel, N., Doerr, J. (2008). Deriving Software Services from Business Processes of Representative Customer Organizations, Service-Oriented Computing: Consequences for Engineering Requirements. International Workshop on Service-Oriented Computing: Consequences for Engineering Requirements SOCCER, 2008 in conjunction with RE 2008, Spain.
- Aier S., Bucher T., Winter R. (2011) Critical Success Factors of Service Orientation in Information Systems Engineering. Business & Information Systems Engineering, Vol. 3: Iss. 2, 77-88.
- ARIS (2006), Help Documentation. ARIS Business Architect 7.0 v. 7.0.2.234414, IDS Scheer AG.
- ARIS Paper, 10 Steps to Business-Driven SOA, ARIS Expert Paper. (2007). Available at: <http://whitepaper.talentum.com/whitepaper/view.do?id=16052>. Accessed August 2010.

- Arsanjani, A. (2004). Service-oriented modeling and architecture. Available at: www.ibm.com/developerworks/webservices/library/ws-soa-design1/. Accessed August 2010.
- Azevedo, L. G., Santoro, F. M., Baiao, F., Souza, J. F., Revoredo, K.; Pereira, V. B.; Herlain, I. (2009a). A Method for Service Identification from Business Process Models in a SOA Approach. In: 10th International Workshop on Business Process Modeling, Development, and Support (BPMDS), Amsterdam.
- Azevedo, L.; Sousa, H. P.; Souza, J. F., Santoro, F. Baiao, F. (2009b). Automatic Service Identification from Business Process Models. IADIS Ibero-American WWW/INTERNET Conference, October, 21-23, Madrid, Spain.
- Azevedo, L. G., Baiao, F., Santoro, F. Souza, J. F., (2011). A Business Aware Service Identification and Analysis Approach. In: IADIS International Conference Information Systems 2011, March, 11-13, Avila, Spain.
- Becker, A., Widjaja, T., Buxmann, P. (2011). Value Potentials and Challenges of Service-Oriented Architectures - Results of an Empirical Survey from User and Vendor Perspective. *Business & Information Systems Engineering*, Vol. 3: Iss. 4, 199-210.
- Bianchini D., Cappiello C., De Antonellis V., Pernici B. (2013). Service identification in interorganizational process design. *IEEE Trans Serv Comput* 7(2):265–278
- Birkmeier, D. Q., Gehlert, A., Overhage, S., Schlauderer, S. (2013). Alignment of business and IT architectures in the German Federal Government: a systematic method to identify services from business processes. In: *Proceedings of the 46th Hawaii international conference on systems sciences (HICSS 2013)*. IEEE Computer Society, Piscataway, pp 3848–3857.
- Bishop, P., Hines, A., Collins, T., 2007. The current state of scenario development: an overview of techniques. *Foresight*, Emerald Group Publishing, Vol. 9, No.1, pp. 5–25.
- BPM-Advisor (2009) Padrões e Notações. Available at <http://www.bpm-advisor.com.br/padnotac.htm>.
- BRG. Business Rules Group. (2001). Defining Business Rules – What are they really? Available at: http://www.businessrulesgroup.org/first_paper/br01c0.htm. Accessed August 2010.
- Cerie, R., Baiao, F., Santoro, F., Discovering Business Rules through Process Mining. In *Enterprise, Business-Process and Information Systems Modeling*, Proc of the 10th International Workshop, BPMDS 2009, and 14th International Conference, EMMSAD 2009, held at CAiSE 2009, Amsterdam, The Netherlands, Lecture Notes in Business Information Processing, 2009.
- DAVIS, R. 2002). *Business Process Modeling with ARIS – A Practical Guide*. London: Springer, pp. 531.

- De la Vara González, J. L. and Sánchez Díaz, J. (2007). Business process-driven requirements engineering: a goal-based approach. In 8th workshop on business process modeling, development, and support (BPMDS'07), Trondheim, Norway.
- Diirr, T., Souza, A., Azevedo, L. G., Santoro, F. 'Analyze Credit Request' Model. (2010). Technical Report, Applied Informatics Department, Rio de Janeiro State Federal University (UNIRIO), RT-0008/2010. Available at: <http://seer.unirio.br/index.php/monografiasppgi>
- Erl, T. (2005). Service-Oriented Architecture: Concepts, Technology, and Design. Prentice Hall.
- Fareghzadeh, N. (2008). Service Identification Approach to SOA Development. In: Proceedings of World Academy of Science, Engineering and Technology, vol. 35, pp. 258-266.
- Gilpin, M. 2010. From The Field: The First Annual Canonical Model Management Forum. Forrester Blogs. http://blogs.forrester.com/print/mike_gilpin/10-03-15-field_first_annual_canonical_model_management_forum
- Gu, Q., Lago, P. (2007). A Stakeholder-Driven Service Life Cycle Model for SOA. In: 2nd International Workshop on Service Oriented Software Engineering: in Conjunction with the 6th ESEC/FSE Joint Meeting, pp 1-7.
- Inaganti, S., Behara G. K. Service Identification: BPM and SOA Handshake, BPTrends, 2007. Available at www.bptrends.com/publicationfiles/THREE%2003-07-ARTBPMandSOAHandshake-InagantiBehara-Final.pdf.
- Jamshidi, P., Sharif, M., Mansour, S. (2008). To Establish Enterprise Service Model from Enterprise Business Model. In: 2008 IEEE International Conference on Services Computing, vol. 1, pp. 93-100.
- Josuttis, N. M. (2007). SOA in Practice: The Art of Distributed System Design. O'Reilly.
- Keller, G, Teufel T., (1998). SAP R/3 Process Oriented Implementation, Addison-Wesley.
- Klose, K., Knackstedt, R., Beverungen, D. (2007). Identification of Services - A Stakeholder-based Approach to SOA Development and its Application in the Area of Production Planning. In: ECIS'07, pp. 1802-1814.
- Ko, R. K., Lee, S. S., and Lee, E. W. (2009). Business process management (bpm) standards: a survey. Business Process Management Journal, 15(5):744–791.
- Kohlborn, T., Korthaus, A., Chan, T., Rosemman, M. (2009). Identification and Analysis of Business and Software Services – A Consolidated Approach. In: IEEE Transactions on Service Computing, Vol. 2, No. 1.
- Le Clair, C. and Teubner, C. (2007). The Forrester Wave: Business Process Management for Document Processes, Q3.
- Marks, E. A., Bell, M. (2006). Service-Oriented Architecture: A Planning and Implementation Guide for Business and Technology. John Willey & Sons Inc.

- McBride, G. (2007). The Role of SOA Quality Management in SOA Service Lifecycle Management. DeveloperWorks. Available at: ftp://ftp.software.ibm.com/software/rational/web/articles/soa_quality.pdf. Accessed August 2010.
- OMG (2011). Business Process Model and Notation (BPMN) version 2.0. <http://www.bpmn.org/>.
- Papazoglou, M. P., Heuvel, W.-J. v. d. (2006). Service-Oriented Design and Development Methodology. In: *Int. Journal of Web Eng. and Tech. (IJWET)*, vol. 2(4), 412-442.
- Papazoglou, M. P., Traverso, P., Dustdar, S., Leymann, F. (2007). Service-Oriented Computing: State of the Art and Research Challenges. *Computer*, vol. 40, no. 11, pp. 38-45.
- Pressman, R.S. (2006). *Software Engineering: A Practitioner's Approach*. McGraw-Hill Science/Engineering/Math, 5th edition.
- Pulier, E., Taylor, H. (2006). *Understanding Enterprise SOA*. Manning.
- Russell, N., ter Hofstede, A.H.M., Edmond, D., van der Aalst, W.M.P. (2004). *Workflow Data Patterns*. QUT Technical Report, FIT-TR-2004-01, Queensland University of Technology, Brisbane.
- Scheer, A.W. (2000). *ARIS - Business Process Modelling*. Springer, Berlin.
- Sharp, A., McDermott, P. (2001). *Workflow Modeling: Tools for Process Improvement and Application Development*. Artech House Computing Library.
- Sun, (2006). Sun SOA enterprise excel. Available at: http://www.sun.com/products/soa/soa_enterprise_excel.pdf>. Accessed in August 2008.
- Systinet (2006). *SOA governance: Balancing flexibility & control within an SOA*. Mercury White Paper.
- Thom, L., Iochpe, C., Reichert, M. (2007). Workflow Patterns for Business Process Modeling. In: 8th Int. Workshop on Business Process Modeling, Development, and Support (BPMDS), pp. 349-358.
- Trkman, P., Kovacic, A., Popovic, A. (2011). SOA Adoption Phases - A Case Study. *Business & Information Systems Engineering* Vol. 3, Iss. 4, 211-220.
- Tsai, W.-T. Wei, X. Paul, R. Chung, J.-Y. Huang, Q., Chen Y. (2007). Service-oriented system engineering (SOSE) and its applications to embedded system development. *Service Oriented Computing and Applications*, pp. 3-17.
- Van der Aalst, W. M. P., Ter Hofstede, A. H. M., Kiepuszewski, B., Barros, A. P. (2003). Workflow patterns. In: *Distributed and Parallel Databases*, vol. 14, pp. 5-51.
- Yin, R.K. *Case Study Research: Design and Methods*. Fourth Edition. SAGE Publications. California, 2009.
- Wall, Q. (2006). Understanding the service lifecycle within a SOA: Design time, Dev2Dev. Available at: dev2dev.bea.com/pub/a/2006/08/.





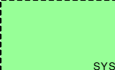







WESKE, M. (2007) Business Process Management – Concepts, Languages, Architectures, Verlag; Berlin; Heidelberg: Springer, pp. 368.

Appendix 1 – EPC notation




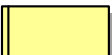







A business process consists of activities coordinated in order to achieve a certain goal. Therefore, it is through the execution of business processes that organizations carry out their purposes (Weske, 2007). The business processes models visually represent aspects of a business process, as flow of activities, performers roles, business rules, information carriers, input and output information, and other information relevant to the process.

This appendix presents the EPC notation (Keller and Teufel, 1998; Scheer, 2000), used to design the business process model “Analyze credit request”. The modeling was created using ARIS¹ tool. Table 11 describes notation’s elements.

Table 11 - Notation of the elements used in process modeling

Nome	Semântica	Sintaxe
(Link to another model)	This symbol indicates that the object as a model associated with it.	
Activity	Represents a process or a step in a sequence that must be performed so that a process be carried out.	
Aspect Activity	Represents a process or a step in a sequence that must be performed so that a process be carried out.	
Automated activity	Represents an activity performed automatically and exclusively on a system without interference of a person. People, equipment, sensors, other systems etc., may interact with this activity just like triggers or receptors of results.	
Automated aspect activity	Represents an activity performed automatically and exclusively on a system without interference of a person. People, equipment, sensors, other systems etc., may interact with this activity just like triggers or receptors of results.	
Database	Represents information or data that is stored in databases of application systems.	
Aspect database	Represents information or data that is stored in databases of application systems	
Data / Information	Represents a set of information (structured or not) generated or consumed during the process execution.	
Aspect data / information	Represents a set of information (structured or not) generated or consumed during the process execution	
End event	Represents the final circumstance or status of the process.	
Star event	Represents a circumstance or status that provides the beginning of the process.	
Intermediate event	Represents a circumstance or status relevant to understanding the process.	

¹ http://www.ids-scheer.com/en/ARIS_ARIS_Platform/3730.html

Process interface	Represents the interface between processes (existing in a VAC), indicating that there is communication between them. In general, it is an indication of another process that complements the flow modeled, but is not the principal object of the model in question.	
Logical operator XOR (exclusive or)	Logical operator representing: - when the split flow: only one of the paths must be traversed, ie, only one destination events must occur. - when they join the flow: just one of the paths traversed starts the next process or activity, ie, just one of the source events must occur.	
Position	Represents the position (role / function) that interacts with a process (producing or consuming information).	
Aspect position	Represents the position (role / function) that interacts with a process (producing or consuming information).	
Business rule	Policy aimed at influencing or guiding the behavior of the business, as support to the business policy that is formulated in response to an opportunity.	
Aspect business rule	Policy aimed at influencing or guiding the behavior of the business, as support to the business policy that is formulated in response to an opportunity.	
Business Requirement	Requirements from the business that will define or restrict aspects of information systems.	
Aspect business requirement	Requirements from the business that will define or restrict aspects of information systems.	
Application system	Represents an information system that supports the execution or executes one or more activities of the process.	
Aspect Application system	Represents an information system that supports the execution or executes one or more activities of the process.	
Organizational unit	Represents an area of the organization (business unit, management, coordination or department) (formal or informal), which interacts with a process.	

1. Value-Added Chain (VAC)

VAC diagram represents organization functions that directly influence the real aggregated value of the organization. Functions can be linked to other functions in order to represent sequence and hierarchy (ARIS, 2006).

The VAC diagram describes the business processes from the more abstract view. Each process model has one or more objectives that add values to guarantee organization life. A VAC model can be detailed in other macro-processes. The highest level value chain represents the organization business process.

Figure 12 presents a VAC diagram example. This model has the macro-process “Manage geophysical processing request” composed by three other macro-process. Coordinated execution of these three macro-processes enables the management of geophysical processing requests.

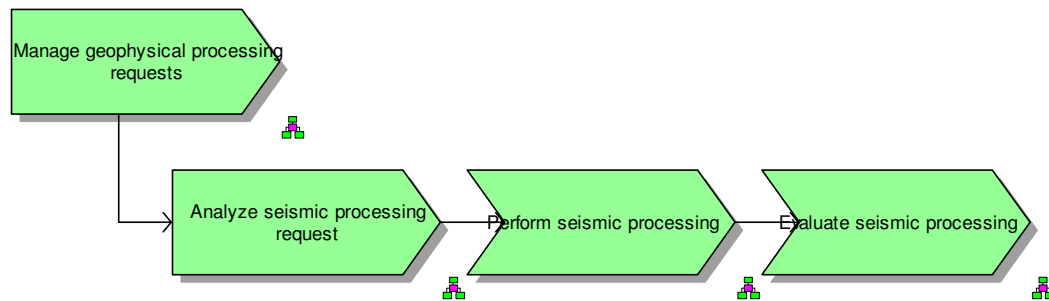


Figure 12 - VAC diagram example.

2. Event-driven Process Chain (EPC)

EPC is the central model in process modeling. It describes a sequence of tasks or activities, that represent the process and adds value to the business (Davis, 2002).

EPC diagram includes flow of activities, roles and organizational units, lanes (according to roles), interfaces to other processes, process start and end events, intermediate events indicating circumstances relevant to the process and logical operators.

Figure 13 presents a EPC diagram example. This model contains activities executed by the interpreter, processing manager, geophysicist and the AIGG system. The activities under the responsibility of each role are respectively in their lanes and the flow between these activities is presented.

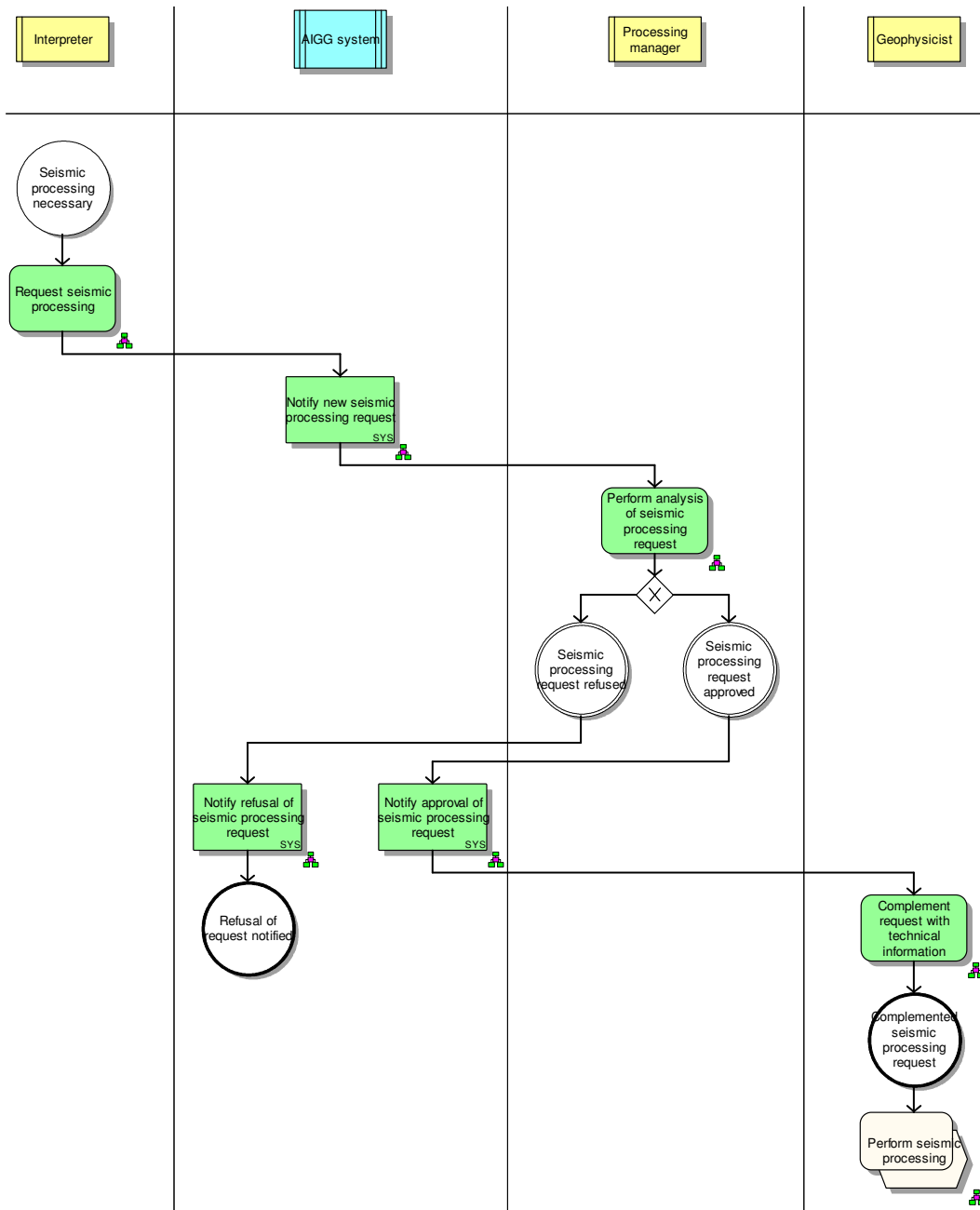


Figure 13 - EPC diagram example

3. Function Allocation Diagram (FAD)

FAD is a diagram detailing a single activity. It is the lowest level model providing a more detailed view of the resources available and needed relevant to an activity (BPM-advisor, 2009).

FAD represents the following activity information: input and output, execution roles, organizational units, systems that support the activity, business rules, business

requirements, indicators, equipments, glossary terms, location, risks etc. Modeling this level of detail depends on the scope of the business process modeling project.

Figure 1 presents an FAD diagram example. This activity is performed by the interpreter, considering the business rule “Seismic processing request”. Input information is “Need for seismic processing” and output information is “Seismic processing request” and “Processing quality expectations”. The system AIGG supports the activity and the business requirements “Register processing request” and “Register quality expectation” are necessary to execute this activity.

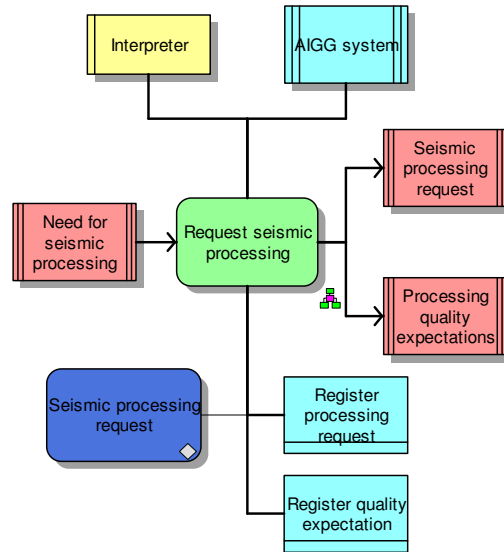


Figure 14 - FAD diagram example

Appendix 2 – Process model “Analyze credit request”

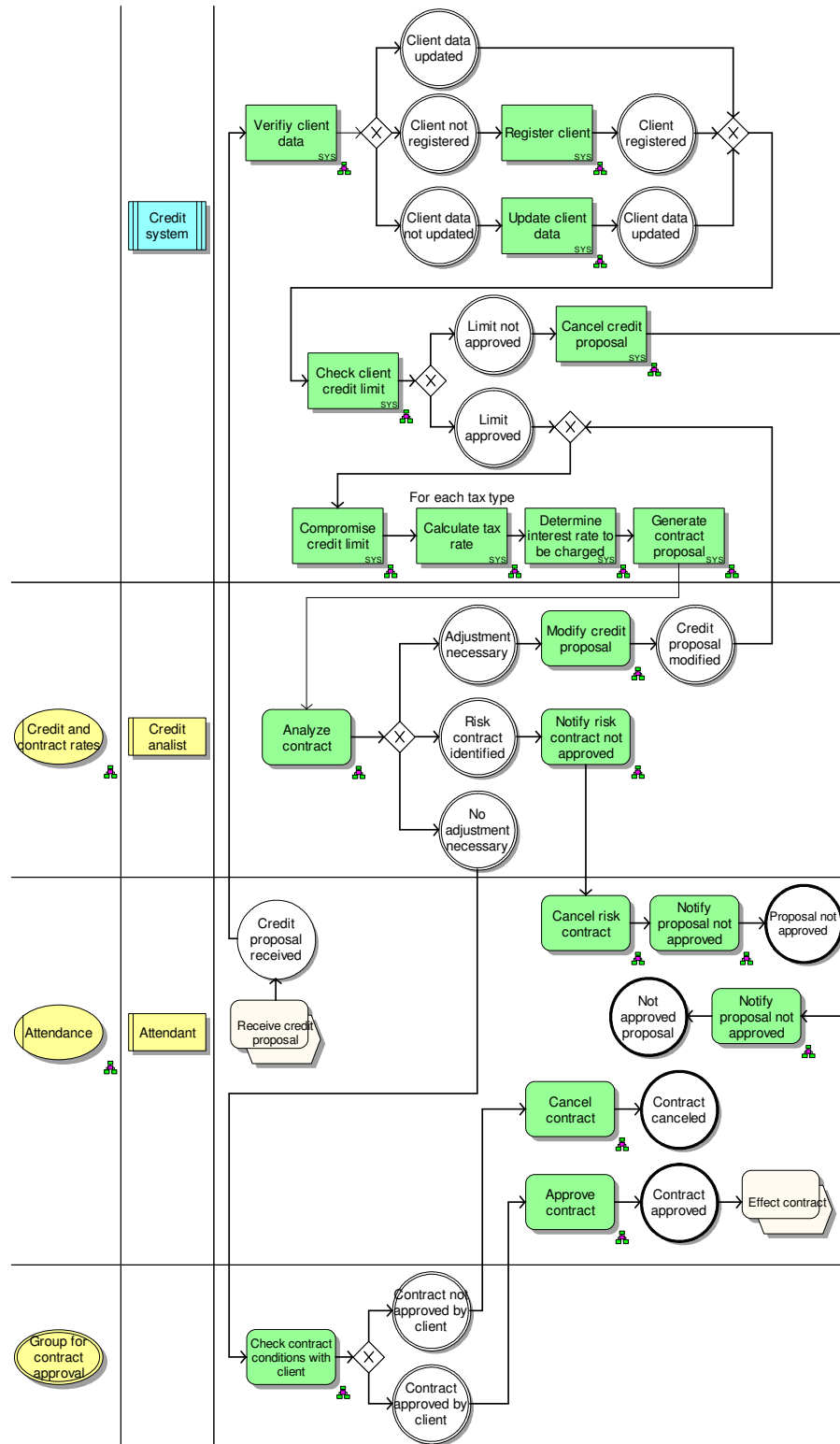


Figure 15. “Analyze Credit Request” process