# MDCSIM: A method and a tool to identify services

Rosane S. Huergo[1], Paulo F. Pires[2]*, and Flávia C. Delicato[2]
[1]Universidade Federal do Rio de Janeiro, Brazil
rosanesfair@gmail.com
[2]CNPq fellow researchers
{paulo.f.pires, fdelicato}@gmail.com

**Abstract**

Service identification is one of the biggest challenges in implementing a service-oriented architecture. Current service identification methods rely on business process descriptions to elicit the business perspective. However, service identification requires a level of business process documentation only found in organizations mature on business process modeling. Besides, current service identification methods have several drawbacks such as the lack of: (i) analyzing both business and IT domains, (ii) identification of both business and software services, (iii) service quality assessment; and (iv) method configurability. In this context, our work overcomes the aforementioned drawbacks by proposing a configurable service identification method (named MDCSIM) that uses master data, logical data models (obtained from organizations databases) and artifact-centric modeling technique. Master data (core enterprise information concepts, needed across different business processes, organizational units and applications across the enterprise) can be used as alternative input to business process. The logical data models aid the identification of master data attributes and contributes to the elicitation of IT perspective and identification of software services. Artifact-centric modeling technique is used along with master data to elicit business perspective and identify business services. MDCSIM also uses some metrics to assess service quality attributes in order to improve the quality of the identified services and of the SIM itself. MDCSIM is supported by a tool named MDCSIM plug-in. Such tool was implemented based on the Model-driven architecture (MDA). MDA enables the transformation of data logical models and artifact-centric models into models that describe candidate services. Finally, an initial assessment of MDCSIM is provided by comparing the service portfolios identified by using MDCSIM and by using other two data-focused service identification methods.

**Keywords:** Artifact-centric modeling; Logical data model; Master data; Model-driven architecture (MDA); Service identification method (SIM); Service-Oriented Architecture (SOA).

## 1 Introduction

The process of service-oriented modeling and design consists of three general steps: identification, specification and realization of services [2]. The identification step aims to determine which services are appropriate to be implemented in a service-oriented architecture. In this phase, services are named candidate services. During the specification step, the service architecture is designed and its interface, messages and events are detailed. Finally, the service is codified and tested in the realization step. The identification step, which is the scope of this work, is one of the major challenges in designing and implementing a service-oriented system [15]. This challenge consists in predicting which services an enterprise will eventually need, and in defining which functions should be part of each service.

*IT CoNvergence PRActice (INPRA)*, volume: 2, number: 4 (December), pp. 1-27
*Corresponding author: Instituto de Matemática, Universidade Federal do Rio de Janeiro, PO Box 68.530, ZIp Code 21941-590, Rio de Janeiro - RJ - Brasil, Tel: 55-21- 3938-8091

During the last decades, several service identification methods were proposed, but there is no consensus on the "best method" nor a predominant approach to identify candidate services. Service identification methods (SIMs) are categorized in three possible identification strategies: Top-down, Bottom-up and Meet in the middle [17]. Top-down strategy identifies services from a business perspective. This strategy commonly uses business process as inputs to identify services [21]. This kind of approach has generally failed to deliver the value promised, because most processes that organizations execute are not enough documented to enable a good service identification [31]. Therefore, this kind of approach requires organizations to be first engaged in modeling business processes before adopting it. Otherwise, service identification effort will be time-consuming and hence will not scale to large business processes [5].

Bottom-up strategy identifies services from an IT perspective. This strategy is driven by functional analysis of existing software assets (applications, services repositories, databases and legacy documentation) to evaluate which functions should be exposed as services. It also identifies new services that can fulfill implementation gaps or meet new requirements and technical specifications [5]. Bottom-up approaches are more successful at delivering services in short-term, but they usually identify fine-grained services that have limited reuse or do not have direct value to the business.

Finally, Meet in the middle is a hybrid strategy that supports the examination of both business and IT perspectives. This strategy is the most aligned with the enterprises reality, since it considers existing software assets and quickly delivers recognizable benefits without neglecting the fact that services are designed for reuse and must be aligned with the business context. Nevertheless, according to current service identification surveys ([7], [8], [10], [21], [25], [34], [36]), only few methods can be classified as Meet in the middle approaches [3], [11], [13], [18], [19], [22], [30], [32], [38]. All of them use processes as inputs to elicit business perspective. Therefore, they suffer from the same drawback of top-down approaches, namely, they require detailed process descriptions as inputs to service identification. Since many organizations are not mature enough in business process modeling, it is important to investigate alternatives to elicit the business perspective.

Besides the lack of Meet in the middle approaches, the analysis of the aforementioned surveys also revealed the following drawbacks in the existing SIMs: (i) lack of identification of both business services and software services, (ii) lack of mechanisms to configure the method according to the organization context (e.g. unavailability of an input, the need to apply the method to small domains or small enterprises) and (iii) lack of mechanisms to assess candidate service quality attributes. In this work, we advocate that Master Data can be useful in this context. Master data is any information considered to play a key role in the operation of a business. Such information encompasses top-level abstraction concepts comprised by data used across different business processes, organizational units, and information systems [16], [28]. These characteristics confer to the master data a great potential of reuse and relevance to the business. Master data can also be correlated with IT perspective by identifying database tables (logical data models) that store their attributes. Hence, master data can be used as an alternative input to business process in service identification methods, aiding the identification of business services and software services from both business and IT perspectives.

Service identification based on master data analysis can be accomplished by modeling tasks and business rules involved in master data transformation (during the master data lifecycle). Therefore, services operations can be modeled as a set of tasks that process master data. This modeling paradigm is used in the Artifact-centric modeling technique [16]. In such technique, an operational model of business processes is defined, in which the lifecycle of business entities (data) is considered as the main driver of the processes. The advantages of this technique are (i) lower modeling effort than in the traditional activity-centric modeling style, since the lifecycle (transformation) of few master data can describe several business processes [6]; (ii) in-depth specification of service operations, through the correlation of the information model and the process model (data lifecycle), in contrast to the activity-centric process

modeling paradigm where it is not mandatory to detail the structure of activities' inputs or outputs [16]; and (iii) the service-oriented nature owned by this model, thus supporting service-orientation design principles such as high level of flexibility, extensibility, and reusability [35].

The usage of master data and artifact-centric modeling technique in a SIM promotes the segregation of capabilities by master data and by the capability type (behavioral or maintenance). This characteristic contributes to the identification of services in layers and therefore supports the customization of the SIM to promote an in-depth analysis of a business domain identifying all layers of services, or a "lightweight" analysis of the business domain to identify only services related to master data maintenance. The identification of services in the scope of distinct business scopes represents the horizontal evolution of the service portfolio. Vertical evolution concerns the identification of services of different layers.

Considering the perspectives that should be comprised in service identification and the aforementioned drawbacks, this work proposes a configurable Meet in the middle service identification method (called MDCSIM) that uses master data, logical data models (obtained from organizations databases) and artifact-centric modeling technique. MDCSIM also uses some metrics to assess service quality attributes of coupling, cohesion, granularity and entity convergence. These metrics are related with SOA goals of promoting business agility, increasing the ROI, and promoting business alignment as detailed in Section 2. The assessment of service quality attributes is important to verify if the candidate services identified by using the method are aligned with SOA goals, and also to improve the quality of the identified services and of the SIM itself.

MDCSIM is supported by a tool named MDCSIM plug-in. Such tool was implemented based on Model-driven architecture (MDA). MDA is designed according to three levels of abstraction models. These models can be used, respectively, to represent the high-level business view, to describe candidate services and to specify service implementation. Models from these three levels are interrelated. One model can be converted to another model through a process named MDA Transformation. The transformation process is supported by standards such as the Unified Modeling Language (UML). MDCSIM plug-in automatically transforms UML diagrams that describe the logical data models and artifact-centric models into models that describe candidate services. The remaining of this paper is structured as follows: Section 2 presents the related work. Section 3 describes MDCSIM and the service layers model used. Section 4 depicts MDCSIM plug-in and the technologies used in its implementation. Section 5 presents the utilization of MDCSIM in a business scenario. Section 6 provides an initial assessment of MDCSIM by comparing the service portfolios identified by using MDCSIM and by using other two data-focused SIMs. Finally, the conclusion is provided in Section 7.

## 2 Related Work

In this Section, we first discuss the drawbacks existing in current service identification methods that brought forth the research gaps exploited in our work, and then we point out which gaps we are addressing, depicting how they are fulfilled.

In the period from 2002 to June 2013, 7 surveys on SIMs ([7], [8], [10], [21], [25], [34], [36]) were identified. Among them, surveys [7], [8], [10], [25] concluded that there is a lack of systematic methods that comprise the identification and analysis of services on both the business and the technical level. The authors in [8], [10], [36] proposed that future methods have to be configurable depending on the utilization constraints within the organizations (e.g. unavailability of an input, the need to apply the method to small domains or small enterprises), while in [21], [25] they suggested that SIMs should analyze non-functional requirements. In [7] the authors remarked that methods do not assess candidate service quality, nor provide means to guarantee it. Finally, in [8] it is suggested that economic aspects should be considered in the identification phase.

Analyzing the aforementioned conclusions, the following research gaps in SIMs were identified:

1. Analysis of both business and technical perspectives: service identification should be wide-ranged and consider multiples perspectives comprising business and technological issues [5]. Business perspective is related to business goals and requirements and it is generally structured into processes or business models that express rules, constraints and dependencies [17]. IT perspective concerns the automation of the business perspective organized into various technology solutions. The analysis of business perspective is important to identify services that deliver direct value to business and promote business agility which is one of the SOA goals [17]. On the other hand, the analysis of IT perspective promotes an alignment with the existing IT assets and helps to identify the resources (data, application functions and existing services) necessary to realize service capabilities, thus providing a better input to the specification phase. These two perspectives are complementary. The analysis of only one perspective can compromise the achievement of SOA goals, or lead to services that are not suitable to the organizations reality. SIMs described in [4], [37] are data-focused methods as MDCSIM, however they analyze only one perspective. SIMs in [3], [11], [13], [18], [19], [22], [30], [32], [38] analyze both perspectives, nevertheless they have disadvantages when comparing with our proposal: (i) the identification of business entities or artifacts is conducted informally, with no theoretical underpinnings [11]; (ii) they fail in providing guidelines to aid the decision of which operations should compose a service [3], [13], [18], [19], [22], [30], [32], [38]; and (iii) they rely on detailed business process descriptions to elicit business perspective, requiring a great effort of documentation processing prior to service identification [3], [11], [13], [18], [19], [22], [30], [32], [38].

2. Identification of both business and software services: this gap is related to the type of candidate service identified by the SIM. There are several different classifications proposed to define service types from various viewpoints. The service classification is often defined based on the value delivered by a service from business and IT perspectives or alternatively based on composition layers. The difference in definitions of service types are based on the scope of each method. Some approaches only provide guidelines to derive services in general [4], [11], [22], [32], [37], [38]; while others distinguish between atomic (basic) services and composite (process) services [3], [13] and a few provide a classification scheme with descriptions of services goals [18], [19], [30]. Services can be separated into business services and software services (IT services) [10]. A business service is an abstraction of one or more business functions or business goals. Software services expose part of an application, perform CRUD operations on databases, or perform functions not related to the business, but required to support business services. Services that perform CRUD operations in databases are often segregated from the IT service type and named Data service (or Informational service) [4], [18], [30]. Regardless of this classification diversity, we concluded that the classification based on the value delivered by a service is more comprehensive (having correspondence with all SIMs) and is the most suitable to analyze whether the SIM identifies services that contribute to promote business and IT alignment, regardless of the service responsibility.

3. Method configurability: SOA implementation is a costly process in terms of time, financial and resources [36]. Many organizations do not involve themselves in implementing SOA, because of the costs associated with the service-oriented modeling process, especially in the service identification phase. Depending on the adopted SIM, several efforts of business process documentation or application reengineering must be taken before adopting it. In order to facilitate SOA adoption, identification methods should be reconfigurable to be compatible to specific situations. For instance, methods should preview the unavailability of an input and suggest alternative inputs. Methods should enable customization to provide a 'lightweight' version for small domains or

cases when fast results are required. None of the aforementioned SIMs provides customization features.

4. Service quality attributes assessment: most researchers agree on the importance of metrics to improve the quality of the identified services and of the SIM itself. Quality is dependent of the stakeholders' requirements, but some general service quality attributes can be identified in a SOA context. Erl [17] emphasizes that the basic software quality design principles of low coupling and high cohesion should be observed during all service creation cycle. Service granularity is also pointed out as a quality attribute, because the granularity level of a service can affect its capabilities, performance, reusability and coupling. None of the aforementioned SIMs assesses service quality, nor makes any effort to improve identified candidates. Regardless of the quality attribute adopted, SIMs should provide means to assess the quality of candidate services . Services with low quality can affect the reuse thus compromising the achievement of SOA goals of promoting business agility and improving the ROI.

5. Elicitation of non-functional requirements: business requirements are not the only requirements that originate candidate services or affect candidate services' capabilities. Non-functional requirements (or technical requirements) might also reveal constraints, conditions of use of a service, or even additional candidate services that support the accomplishment of non-functional requirements. For instance, non-functional requirements of security can originate services to authenticate users, to control the access to specific functionalities, or to limit access to some services depending on the user's profile. Besides this, conflicting non-functional requirements might cause service redesign [21] impacting the whole process of service-oriented modelling and design. SIMs might elicit non-functional requirements by using Service Oriented Design Aspect (SODA) technique [10] or have an activity to elicit non-functional requirements [18]. SODA technique identifies services based on the decomposition of interactions, concerns and features into aspects and composing them according to requirements [12].

6. Analysis of economic aspects: technically-driven implementations often fail to be profitable [8]. Deployment of certain services can promote business advantages such as reducing the time-to-market of a new product, decreasing maintenance and operation costs by reducing IT complexity and decreasing vendor dependency. These analyses are important to economically advantageous implementation of business services. The customer has to be willing to pay for the result of a process, i.e. services should always increase the value of a product. The degree of value creation depends on an effective and efficient combination and coordination of resources [8]. None of the aforementioned SIMs analyzes economical aspects.

Among the aforementioned research gaps, three are addressed by MDCSIM: *Analysis of both business and technical perspectives, Identification of both business and software services* and *Service quality attributes assessment*. The research gaps *Elicitation of non-functional requirements* and *Economical aspects analysis* are suggestions for future research in service identification field.

The *Analysis of both business and technical perspectives* is accomplished by using Master data and their lifecycle to describe the business perspective. Master data lifecycle depicts tasks and business rules involved in master data transformation. Each master data transformation can be correlated with several business process activities in the traditional activity-centric modeling paradigm, thus describing the business domain. Master data is composed of a set of attributes that describe it. For example, the attributes Value, Payment date and Currency are part of the master data Payment. Master data attributes and the tables where they are stored in can be identified from logical data models, complementing the business perspective identified previously with the IT perspective.

The analysis of both perspectives is also related with the identification of business services and IT services. Business services are identified from tasks and business rules within master data lifecycles and IT services from the logical data model obtained from the existing databases. MDCSIM uses a service layer model in order to aid service identification. These layers can be organized accordingly to the value delivered to business as described in Section 3.

The gap *Service quality attributes assessment* is addressed by the evaluation of the granularity, coupling, cohesion and entity convergence quality attributes of the identified service portfolio. These quality attributes were chosen because they are related with SOA goals of promoting business agility, increasing the ROI, and promoting business alignment [17]. Business agility can be favored by loosely-coupled services, because a service with low coupling is self-contained and independent, thus it is more easily reused and composed in order to support new requirements. Cohesion and entity convergence cover the ROI increasing. Cohesion increases the comprehension of identified services, thereby simplifying reuse, maintenance and future enhancements. Services that encapsulate all actions of a business entity are more reusable, avoiding redundancy. Granularity covers the business alignment goal, because coarse-grained services offer rich functionality and have larger contribution to business processes. The evaluation of the aforementioned quality attributes is accomplished by using metrics proposed in [27]. These metrics are detailed in Section 6.

The gap *Method configurability* is partially addressed, because MDCSIM enables a customization of its steps in order to provide a "lightweight" version, but it does not enable utilization of alternative inputs when the list of master data or the master data lifecycle models are not available. The complete version of MDCSIM supports the identification of services for distinct business scopes, which is the horizontal evolution of the service portfolio and the identification of services of different layers, which corresponds to the vertical evolution. The "lightweight" MDCSIM version enables short iterations in order to identify first a set of services that perform CRUD operations in databases (Entity services) for the prioritized business scopes (horizontal evolution). The "lightweight" version can be used by organizations with low maturity in SOA or to quickly deliver results.

## 3 MDCSIM: Master Data-Centric Service Identification Method

This section introduces MDCSIM and depicts its steps, artifacts, roles and a service layer model used to aid the service identification.

In order to make services identification easier, services can be categorized into abstraction layers [1]. Layers are related to the service reuse potential and to the logic encapsulated by them. The organization in layers is often used to guarantee the definition of services with "right" granularity, cohesion and it is also a natural composition hierarchy. This work uses the service layer model proposed by [17], [25] to guide service identification. The model defines four layers of services: Utility services, Entity services, Task services and Process services. Some layers' definitions have a great correlation with business processes and activities, referring to the traditional activity-centric process modeling paradigm. Thus, these definitions were extended to comprise also services identified from the artifact-centric process modeling paradigm using state transitions, as follows.

Utility service is not related to business logic. It provides shared functions for other services, such as authentication, encryption, logging and event handling [17], [25]. Entity service is a business service whose functional scope is related to the functional context of one business entity [4], [17], [24], [25], [26]. In MDCSIM, an Entity service manipulates one master data and ensures data completeness. Task service is modeled for specific processes to meet immediate requirements of the organization and therefore it contains specific business logic [4], [17], [24], [25]. It represents the behavior of a business entity and implements the transitions in a master data lifecycle [31], [33]. Finally, Process service introduces a

level of abstraction that alleviates the need for Entity, Task, and Utility services to manage interaction details required to ensure that service operations are executed in a specific sequence. A process service represents a workflow of set of states of one entity or of a set of state transitions of several entities. The four layers, their granularity and potential for reuse are presented in Figure 1.

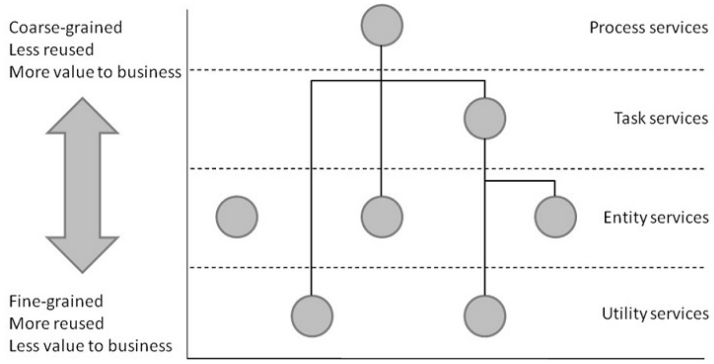


Figure 1: Service Layers. Circles represent services and and lines connecting the circles represent service dependencies.

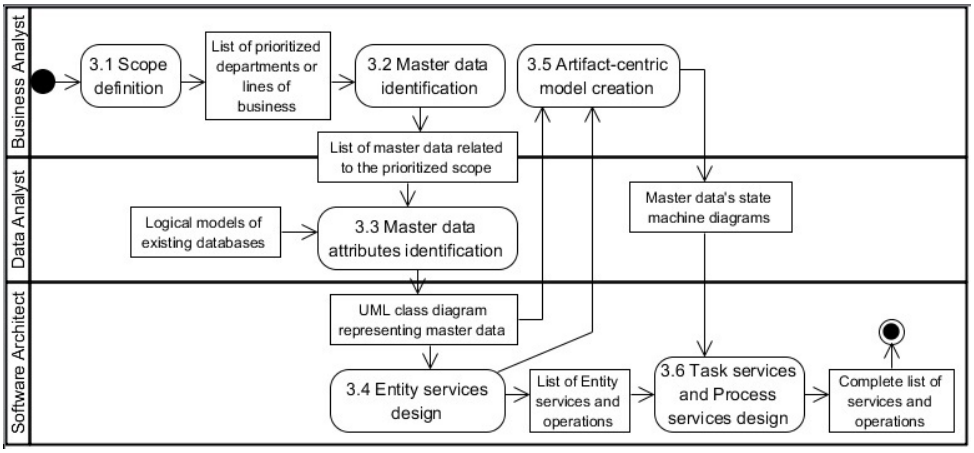


Figure 2: MDCSIM steps.

MDCSIM intends to support the identification of Entity services, Task services and Process services. Nevertheless, Utility services can also be identified by analyzing cross-cutting concerns not related to business in Entity services, Task services and Process services and exposing them as Utility services. The identification of Entity services CRUD operations uses a subset of the operation patterns proposed in [4]. MDCSIM also uses data models as described in [4]. Nevertheless, these models are correlated with master data to aid the identification of business relevant concepts. Some guidelines to aid master data identification were also provided as part of our proposal. The UML activity diagram of Figure 2 presents the steps of MDCSIM, which are detailed in the next sections.

### 3.1 Scope definition

Service identification cannot be applied in all organizational units at the same time. Instead, it makes sense to prioritize the analysis at core lines of business or departments, in which a number of visible benefits and the potential of service reuse can be estimated. Thus, the first step of MDCSIM is the

definition of the scope definition. Scope definition step is conducted by the Business Analyst and it produces as output a prioritized list of lines of business or departments.

The scope can be delimited by choosing a line of business or a department. Within large organizations, lines of business frequently act as sub-organizations. Therefore, it could still be necessary to restrict the scope to a department. Subsequent iterations of the method should be done in lines of business or departments that interact with the ones previously prioritized, because several master data will be shared between them. Therefore, Entity services already identified will be potentially reused.

It is important to mention that our method supports the horizontal and vertical evolution of the service portfolio. Horizontal evolution is related to the identification of services in the scope of distinct lines of business or departments. Vertical evolution concerns the identification of services of different layers, from the fine-grained to the coarse-grained, accordingly to the aforementioned layers. This is an important feature since the method can be customized to support service identification in organizations with different levels of SOA maturity. Organizations can start with the identification of Entity services of several lines of business or departments executing the steps 3.1 to 3.4. Later, as the maturity in SOA increases, the identification of Task and Process services can be accomplished by executing steps 3.5 and 3.6.

### 3.2 Master data identification

The next step is the elicitation of a set of master data that are relevant to the scope prioritized in the previous step. Common organization master data can be elicited from the following domains [28]:

- Parties: Roles played by persons or organizations, such as patients, suppliers and employees.
- Things: Products, services or other items used in the production lifecycle (from development, through manufacturing, sale and delivery).
- Account: Addresses how a Party is related to a Thing. Cost centers and contracts are examples of master data in the account domain.
- Location: Usually is associated with the aforementioned domains. Geographic position is an example of location master data.

Master data can be identified by answering the following questions: “Who?”, “What?”, “How?” and “Where?”. Question “Who?” addresses the Party domain. The question “What?” addresses the Things domain. The question “How?” addresses the Account domain and “Where?” the Location domain.

The step of master data identification is conducted by the Business Analyst with participation of the Data Analyst. The output is a list of master data related to a business scope.

### 3.3 Master data attributes identification

This step aims to identify master data attributes and relationships by correlating the master data identified in the previous step with the logical data model that describes them. This correlation promotes a systematic way to identify which attributes should compose a master data. The logical data models can be gathered by using relational database reverse engineering [12], Latent Semantic Indexing (LSI) technique [9] or master data catalogues within the organization. Logical models are presented as UML class diagrams. The logical models must be manually analyzed by the Business Analyst and the Data Analyst in order to identify which classes describe the master data identified in the step 3.2. Each master data should be described by one class. Classes that describe master data are marked with the stereotype “MasterData” by the Data Analyst. The logical model must be normalized to derive classes whose attributes

were aggregated using the functional dependency constraints. Therefore, the master data are comprised by a cohesive set of attributes. The master data cohesion affects the resulting service cohesion, because a service will act in a set of attributes defined by each master data.

This step is conducted by the Data Analyst with participation of the Business Analyst. The output is a stereotyped UML class diagram representing the master data, their attributes and the classes that have a relationship with them.

### 3.4 Entity services design

After identifying master data and their attributes, a set of Entity services and basic operations can be designed. Basic operations are CRUD operations that act on master data attributes.

Each class marked with the stereotype "MasterData" in the UML class diagram produced in the previous step originates one Entity service with the same name. This rule ensures cohesion of the Entity services because they deal with a set of attributes grouped by functional dependency constraints. Entity service operations should follow the format described in Table 1. This format is a subset of the CRUD operations defined in [4]. This subset was chosen based on the needs of composition to create the coarse-grained Task services and Process services as described in the step 3.6.

Table 1: CRUD operations format.

| Operation | Services specification | Explanation |
|---|---|---|
| Create | Boolean CreateMasterDataName (String tn, Object la, Object lv) [a] | When inserting a row that contains values for foreign keys, check if it exists as value of a primary key. |
| Update | Boolean UpdateMasterDataName (String tn, Object la, Object lv, String wc) | When updating a list of attributes that contains values for foreign keys, check if it exists as value of a primary key. |
| Delete | Boolean DeleteMasterDataName (String tn, String wc) | When deleting a row check if it is referenced as foreign key by other tables. |
| Read | Object ReadMasterDataName (Object lt, Object la, String wc) | Retrieves specific attributes from one or more tables. |

[a] tn - table name, lt - list of tables, la - list of attributes, lv - list of values, wc - where clause

This step is conducted by the Software Architect with participation of the Data Analyst. The output is a list containing all Entity services and methods elicited in this step.

### 3.5 Artifact-centric model creation

Artifact-centric process modeling technique defines business processes in terms of interacting business artifacts. Each business artifact is characterized by an information model (set of attributes) and a lifecycle model describing tasks that can be invoked on these artifacts [14]. A service comprises one or more tasks that perform operations on some artifact(s), where these operations should reflect steps of progress towards the business goal.

In MDCSIM, the master data play the role of artifact whose information model is expressed by the UML class diagram defined in the step 3.3. The lifecycle model is expressed by UML state machine diagrams created in this step. State machine diagrams are used to identify Task and Process services that implement business rules, interactions among master data or other activities that use Utility or Entity services. A state machine diagram must be constructed for each class marked with the stereotype "MasterData". Transitions and business rules must be modeled using the Events, Conditions and Actions

format and OCL (Object Constraint Language) [1]. Each transition is an operation in a Task service or an Entity service. A set of transitions that are executed in a predefined order can be modeled as a Process service. Rules to transform state machine transitions into services are detailed in the step 3.5.

This step is conducted by the Business Analyst and the Software Architect. The output is a set of UML state machine diagrams.

### 3.6 Task services and Process services design

This step aims to update Entity services operations and to design Task services and Process services. Each state machine diagram constructed in the previous step has its transitions transformed into service operations. Each transition is mapped to one operation. This rule states that an operation comprises the events, conditions and actions modeled into one transition. Basic CRUD operations can be referenced within events, conditions and actions. In this case, the identified operation will reference an Entity service. The master data attributes manipulated by the transition must also be included as operation parameters.

Operations should be grouped into services in accordance with the pattern presented in Table 2. This pattern analyzes the source and target states of the transition that originated the operation and defines the destination service where the operation will be included. This pattern aims to segregate transitions related to the master data behavior from transitions related to master data maintenance. The application of this pattern also ensures that services are identified according to the layers model (Figure 1). After identifying

Table 2: Grouping operations pattern.

| Condition | Destination service |
|---|---|
| The transition source state is the same of target state. | This operation should be included in the Entity service that represents the master data that owns the state machine diagram where the transition was modeled. |
| The transition source state is different from the target state. | This transition is related with control and should be included in a Task service. It should be created only one Task service per state machine diagram to gather operations derived from this kind of transitions. |

Task services and updating Entity services, the Software Architect can analyze the need to represent sequences of state transitions as Process services. The transitions that participate in orchestrations must be marked with the stereotype “Orchestration”. This step is conducted by the Software Architect. The output is the service list comprising the updated Entity services and the Task services, Process services and Utility services identified in this step.

## 4 MDCSIM plug-in implementation

This section provides an overview of the model-driven approach justifying its utilization to automate MDCSIM. MDCSIM plug-in is presented and its operation, inputs and outputs are depicted.

According to the Object Management group (OMG) the model driven architecture (MDA) is an approach to use models in software development. MDA separates the specification of a system from the details about the way that system uses the capabilities of its computational platform [29]. MDA is designed according to three levels of abstraction: the Computation Independent Model (CIM), the Platform Independent Model (PIM) and the Platform Specific Model (PSM). CIM models focus on requirements and business rules of the systems. PIM models describe the operation of a system independently of a target platform. PSM complements information of the PIM models with an additional focus on a specific platform to be used by the system.

Models are interrelated. One model can be converted to another model of the same system in a process named Transformation. The transformations can be Model-To-Model or Model-To-Code. The process of transformation is supported by standards such as Meta-Object Facility (MOF), Unified Modeling Language (UML) and XML Metadata Interchange (XMI) in order to ensure high level of completeness and consistency of the models. MOF is a standard used in the specification and development of meta-models. A meta-model precisely describes the properties and constructs of every model. UML, as a graphical modeling language, provides the basic constructs to define and visualize meta-models. XMI defines rules for interchanging models.

In service-oriented analysis and design the high-level business view can be represented with CIM models, while the information system view can be represented first by PIMs, and then specified in PSMs [10]. For the service identification phase, CIMs are used to describe business domain and PIMs are built to identify candidate services, while PSM can be used in the service specification phase. Since the models defined in MDA can be correlated to the process of service identification and MDCSIM uses UMLs diagrams to describe the business domain, MDA is a natural choice as the underpinning technology for the implementation of a tool to support MDCSIM (MDCSIM plug-in). Such tool aims to facilitate MDCSIM utilization through the automation of its steps. Besides guiding the user across the method, the tool also allows the manipulation of business domain models with large number of entities thus, contributing for the scalability of MDCSIM.

MDCSIM plug-in is a tool that reads the stereotyped UML class diagram representing the master data created in the step 3.3 and the UML state machine diagrams that represent master data lifecycles created in the step 3.5 and applies the rules defined in the steps 3.4 and 3.6 in order to generate a UML class diagram of identified candidate services. MDCSIM plug-in builds on MDA abstractions. The CIM model corresponds to the master data UML class diagram and UML state machine diagrams and the PIM model corresponds to the UML class diagram of identified services. The PSM abstraction is not used in the identification phase. Nevertheless, MDCSIM plug-in can be extended to generate PSM models supporting the specification and implementation phases in the service design process.

Figure 3 shows MDCSIM plug-in inputs, outputs and the correlation of each element with MDA abstractions.

MDCSIM plug-in was built using ATLAS transformation language (ATL) [23]. ATL provides a modeling transformation platform to transform a set of source models into a set of target models. An ATL transformation is composed of rules that define how source model elements are navigated to create the elements of the target models. The source and target models conform to meta-models or standardized meta-meta-models such as MOF and Ecore. ATL is developed on top of the Eclipse environment as an Integrated Development Environment (IDE) and an ATL transformation engine is used to compile and execute ATL programs. ATL was chosen because it provides a complete transformation model and supports complex transformations. Moreover, ATL has gained extensive support for development from the user community (various examples and case studies are available).

The source and target models were constructed and visualized using Papyrus [20]. Papyrus is a graphical editing tool for UML2 as defined by OMG. As ATL, Papyrus is developed on top of the Eclipse environment as an IDE. Papyrus has also the advantage to offer advanced support for UML profiles. The source and target models comply to a profile meta-model which is based on Ecore meta-meta-model. The profile defines two stereotypes: *MasterData* and *Orchestration*. *MasterData* stereotype is a Class stereotype. It is used to identify which classes correspond to a master data in the class diagram used as input for MDCSIM plug-in. *Orchestration* is a Transition stereotype. It is used in the state machine diagrams in order to identify which transitions are parts of orchestrations. *Orchestration* stereotype has an attribute named *serviceName* that defines the name of the service and of the operation that orchestrates the transition(s). For instance, when two transitions have the stereotype *Orchestration* and the same name in the attribute *serviceName* they are part of the same orchestration. One transition can participate

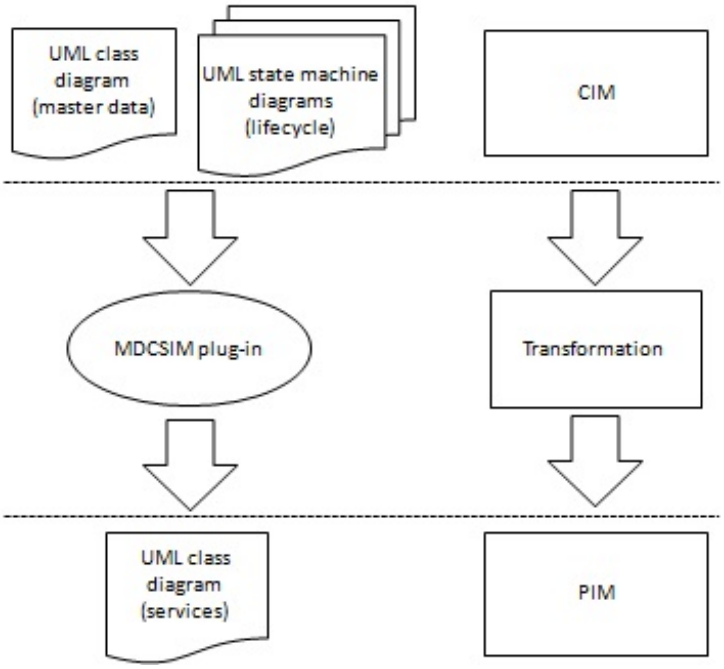


Figure 3: MDCSIM plug-in correlation with MDA abstractions.

of more than one orchestration. Orchestrations originate Process services.

The utilization of MDCSIM plug-in, as well as the construction of the aforementioned source models are demonstrated in the Section 5.

## 5 Service identification using MDCSIM

In this section, MDCSIM is applied to a real business scenario in order to show the practical usage of the aforementioned steps and the MDCSIM plug-in.

The business scenario used to test MDCSIM is of a reinsurance company. The core areas of the reinsurance company are Sales and Claim processing. The Sales area comprises the activities of risk analysis, underwriting and premium processing. The Claim processing area comprises the activities of claim notification receipt, claim analysis (regulation), and claim recovery. The Claim area was prioritized because it deals with several legal constraints (e. g. time to answer claim notifications or indemnities payments). Within the scope of the Claim area, two iterations of MDCSIM will be performed in order to demonstrate service reuse between iterations.

The first process of the Claim area is Claim Notification Reception. The Claim reception department receives claim notifications and analyzes them in order to identify: (i) if the customer loss is covered by the contracted covers in the reinsurance, (ii) if the reinsurance is up to date, or (iii) to calculate the applicable penalties when the notification is not sent in the proper period. The notification can be reproved or forwarded to the claim recovery department. If it is forwarded, a provision to liquidate the claim must be created, the contractor must be blocked and the foreign companies that participate in the reinsurance as well as the managers must be notified depending on the estimated loss value. Figure 4 shows the activity diagram of the Claim Notification Reception process. It is important to note that the business process of Figure 4 (and Figure 5) are not used as an input of MDCSIM, the purpose of such process documentation is solely illustrative. The second process of the Claim area is Claim Recovery process. The Claim can be regulated by the own reinsurer or delegated to the risk participant

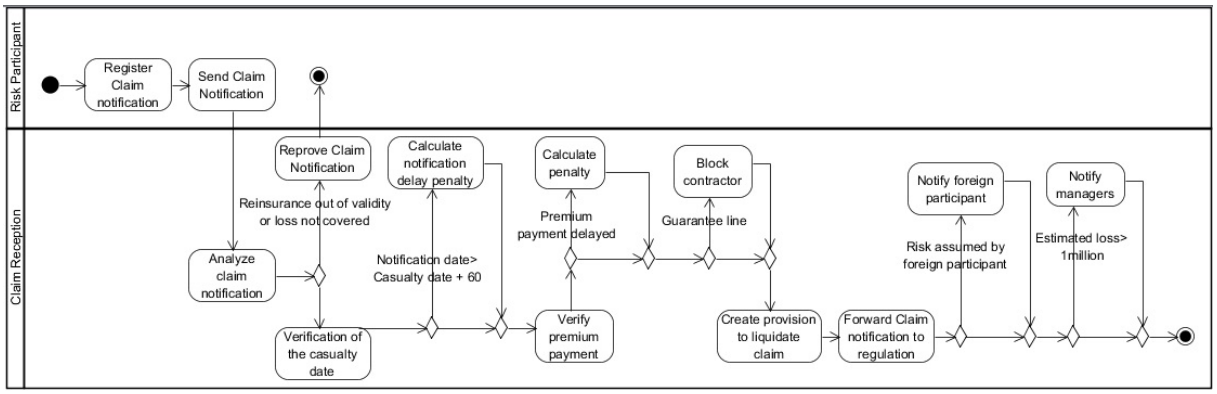


Figure 4: Claim Notification Reception process.

depending on the percentage of reinsurance assignment and on the estimated loss value. When the claim is regulated by the risk participant, the Claim recovery department is responsible to monitor the process and answer to the participant requests. Otherwise, the Claim recovery department defines a regulator to deal with the claim. The regulator performs inspections in order to calculate the real loss and confirm the accident cover. The regulator can ask the Claim recovery department to adjust the claim provision, to refuse the claim or to authorize the claim recovery. Accordingly to the regulator request, the Claim recovery department updates claim entries (provisions, expenses or indemnities paid for the claim), registers payments, authorizes claim recovery or refuses claim recovery and notifies the risk participant. The risk participant can send recovery requests (requests for indemnity anticipation) which are also analyzed by the claim recovery department. Figure 5 shows the activity diagram of the Claim Recovery process.

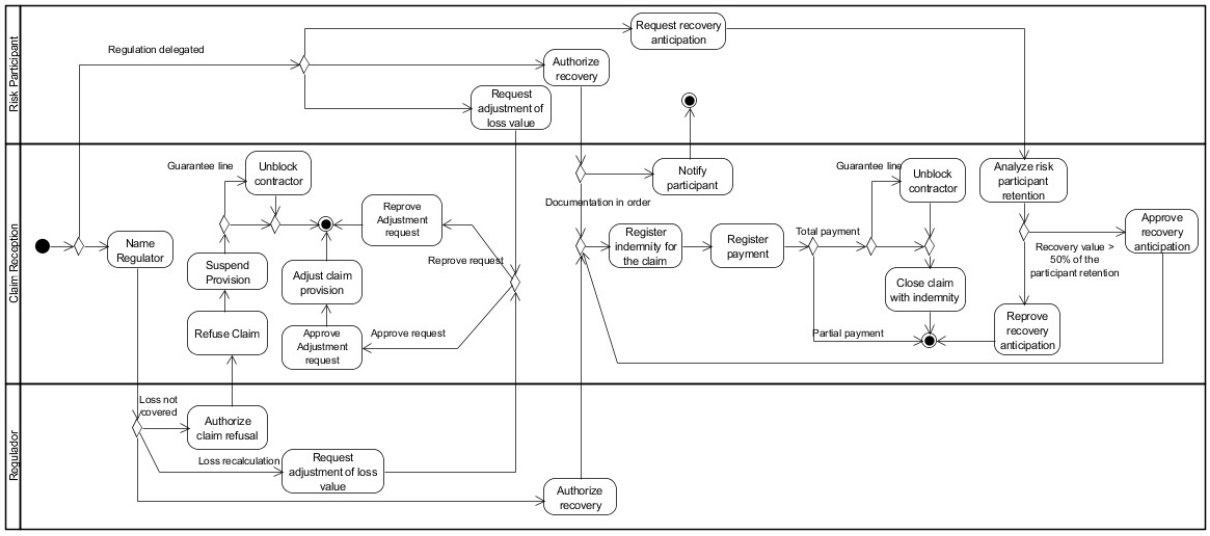


Figure 5: Claim Recovery process.

## 5.1 First iteration - Claim Notification Reception process

After defining the scope of analysis as stated in the step 3.1, the related master data should be identified. Master data identification is supported by the questions described in step 3.2. The master data Contractor,

Reinsured and Risk participant corresponds to the Party domain (identified by the “Who?” question). The master data Cover corresponds to the Things domain (identified by the “What?” question). Finally, Claim entry, Claim notification, Contract, Line, Modality, Premium and Reinsurance address how the Reinsured is related to the Reinsurer or the Risk Participant is related to the Reinsurer (identified by the “How?” question). In order to identify master data attributes as stated in step 3.3, the UML class diagram presented in Figure 6 was constructed. The master data class diagram was constructed by consolidating master data definitions provided by business specialists and the logical data models obtained from reverse engineering of the databases used by the systems that support Claim Notification Reception process.

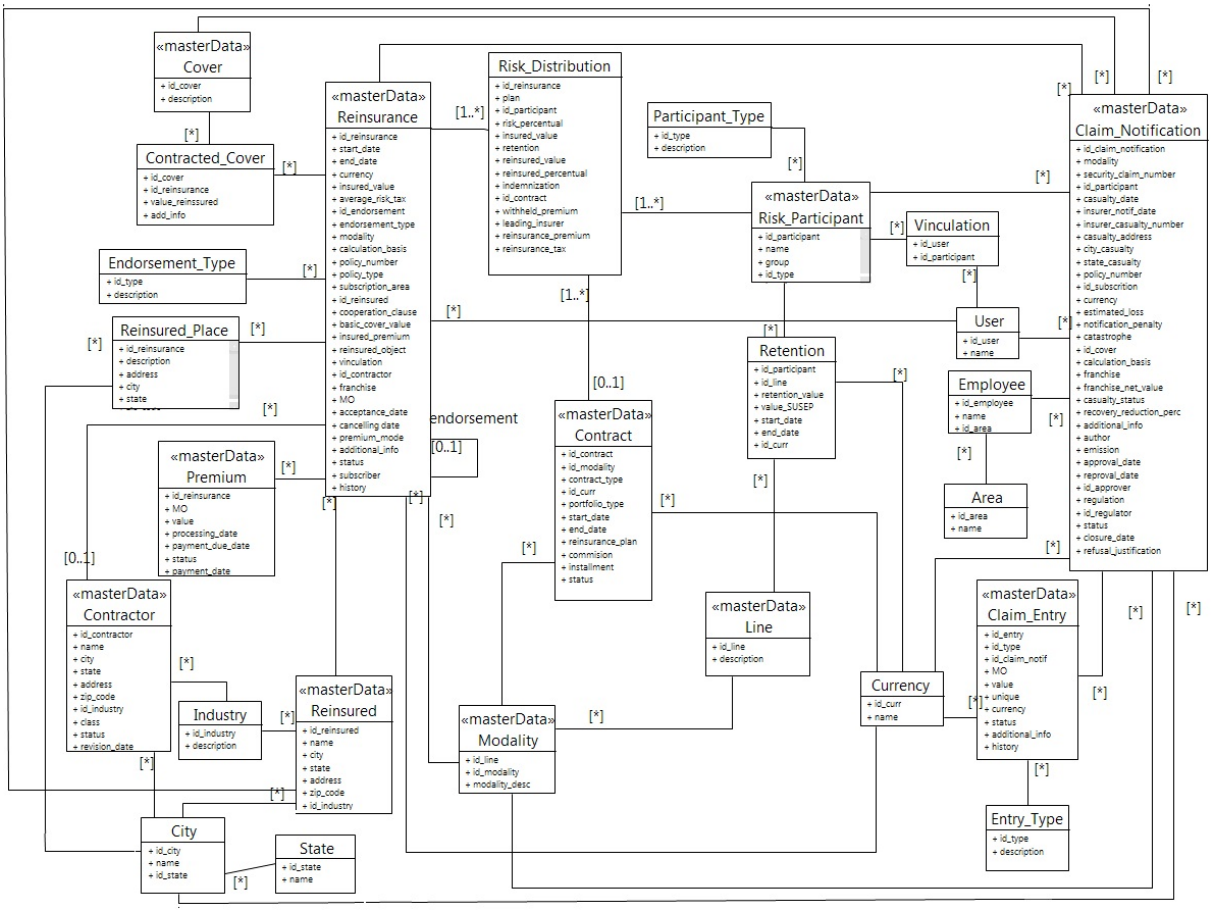


Figure 6: UML class diagram representing the logical data model.

The diagram was modeled with the aid of MDCSIM plug-in. The classes that represent the aforementioned master data are marked with the stereotype masterData and named with the corresponding master data name. Figure 6 presents only the main attributes of the business domain in order to be legible. Nevertheless all attributes that compose a master data are taken into consideration by the CRUD operation patterns presented in Table 1.

In the next step of MDCSIM (3.4) each master data of the UML class diagram presented in Figure 6 originates one Entity service that comprises the CRUD operations presented in Table 1. This task was accomplished by running MDCSIM plug-in using as input the master data class diagram as presented in Figure 6. The Entity services *Claim_Notification, Claim_Entry, Contract, Contractor, Cover, Line, Modality, Premium, Reinsurance, Reinsured* and *Risk_participant* were identified. The classes that represent these services are presented in Figure 8.

The next step (3.5) is the Artifact-centric model creation, when state machine diagrams must be designed to master data identified in the previous steps. Master data that do not have transitions different from the CRUD operations do not need a state machine diagram (in this scenario, this was the case for Cover, Line, Modality and Reinsured). The state machine diagrams were modeled with the aid of MDCSIM plug-in. An example is presented in Figure 7. Some basic CRUD operations (for example read and delete operations) were not included in the state machine diagrams, in order to simplify the visualization. Nevertheless, this kind of operation was already identified in the step 3.4. Transitions that must be orchestrated in order to compose Process services were marked with the stereotype *Orchestration* and the name of the Process service was defined as a property of the stereotype. In the last step of MDCSIM (3.6), each

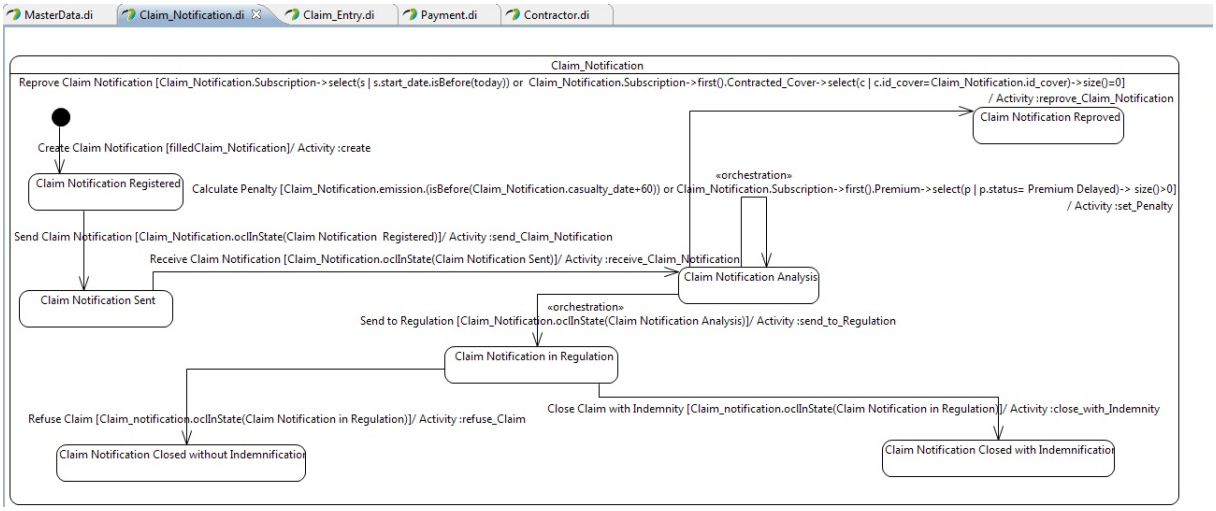


Figure 7: State machine diagram of the Claim Notification master data lifecycle.

transition that does not correspond to a basic CRUD operation identified in the step 3.4 was transformed into a service operation. Services operations were identified by running MDCSIM plug-in using the state machine diagrams designed in the previous steps as inputs. Operations were included in Entity services identified previously or in Task services created in this step according to the pattern described in Table 2. Each created Task service was named adopting the convention <Master data name_Controller>. For example, the transition */setPenalty (int id_claim_notification)* does not change Claim notification's state, so it becomes an operation with the same name as the transition in the *Claim_Notification* service. On the other hand, the transition */reprove_Claim_Notification(int id_claim_notification, date reproval_date)* was added to the *Claim_Notification_Controller* service, because it changes the Claim Notification's state to "Claim Notification Reproved".

MDCSIM plug-in also created process services by identifying transitions marked with the stereotype Orchestration and gathering them into the Process service whose name was defined as the stereotype property. A total of 19 services were identified: 11 Entity services (*Claim_Entry, Claim_Notification, Contractor, Contract, Cover, Line, Modality, Premium, Reinsurance, Reinsured and Risk_Participant*), 6 Task services (*Claim_Entry_Controller, Claim_Notification_Controller, Contract_Controller, Contractor_Controller, Premium_Controller* and *Reinsurance_Controller*) and 2 Process services (*Analyze_Claim_Notification_Process* and *Create_Endorsement_Process*). *Analyze_Claim_Notification_Process* service orchestrates the services *Claim_Notification, Claim_Notification_Controller, Contractor_Controller* in order to calculate the applicable penalties, block the contractor and then send the claim notification to the regulation department. *Create_Endorsement_Process* service orchestrates the *Reinsurance* and the *Reinsurance_Controller* services in order to expire the previous reinsurance when an endorsement is created.

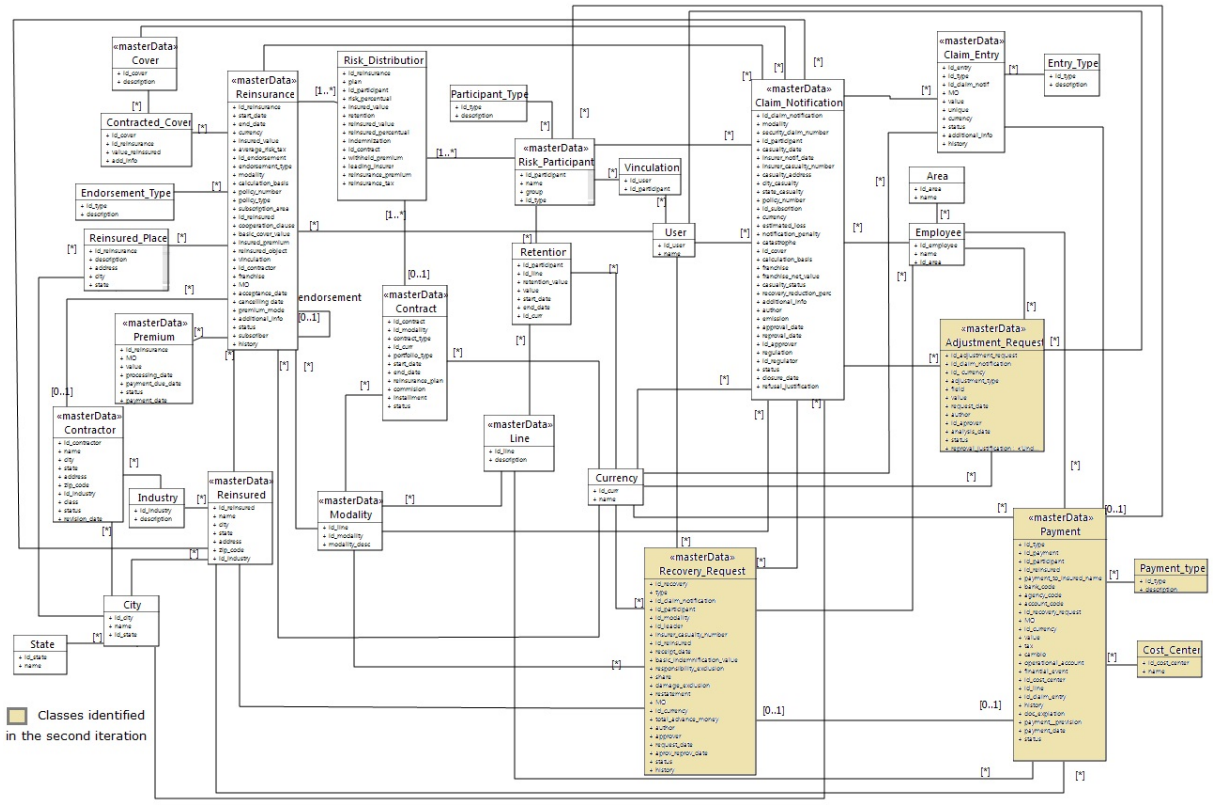


Figure 8: UML Class Diagram representing the updated logical data model.

The aforementioned services are presented in Figure 9.

## 5.2 Second iteration - Claim Recovery Process

The prioritized process for the second iteration was the Claim recovery process. For this process, 3 master data were identified. The master data Adjustment Request, Payment and Recovery Request address how the Risk Participant is related to the Reinsurer (identified by the "How?" question). Claim entry, Claim notification, Contractor, Reinsured and Risk participant were already mapped in the previous iteration. In order to identify master data attributes, the UML class diagram constructed in the previous iteration was updated with the master data and the classes that have relationship with them identified in this iteration as presented in Figure 8. For the next step (3.4) MDCSIM plug-in was executed using as input the updated master data class diagram presented in Figure 8. Three new Entity services were identified *Adjustment_Request, Payment* and *Recovery_Request*.

In the step 3.5, state machine diagrams were designed for the master data identified in this iteration (Adjustment Request, Payment and Recovery Request). The state machine diagrams are similar to the diagram presented in Figure 7. The state machine diagrams modeled in the previous iteration were also revisited, in this step, in order to identify new transitions, or transitions that must be orchestrated to fulfil requirements of the Claim recovery process. No new orchestrations were identified, but a transition (Name Regulator) was included in the Claim Notification diagram. These diagrams were processed by MDCSIM plug-in in order to identify Task and Process services as stated in the step 3.6. A total of 6 new services were identified: 3 Entity services (*Adjustment_Request, Payment* and *Recovery_Request*) and 3 Task services (*Adjustment_Request_Controller, Payment_Controller* and *Recovery_Request_Controller*).

The final list of Entity, Task and Process candidate services identified for both iterations is presented in Figure 9.

Among the services identified in the first iteration, 9 were reused in the second iteration (*Claim_Entry, Claim_Notification, Claim_ Notification_Controller, Contractor_Controller, Line, Modality, Reinsurance, Reinsured* and *Risk_ Participant*). Providing reusable services is an important feature of SIMs in order to accomplish SOA goals of increasing organizational agility and ROI. Another important remark is that some operations of the Task services were identified during the elicitation of the master data lifecycle in the first iteration but they are not required in the scope of the Claim Reception process. However, they were required within the scope of another business processes. This is the case of operations *suspend_Claim_Entry* of the *Claim_Entry_Controller* service, *close_with_Indemnity* and *refuse_claim* of the *Claim_Notification_Controller* service and *unblock_contractor* of the *Contractor_Controller* service that were used in the second iteration ( Claim recovery process) as they were modelled, that is,without any refactoring the original service interface. This feature provides more stable candidate services, with interfaces that do not need to be changed in order to support new business areas.

# 6 Assessment of MDCSIM

In order to provide an initial assessment of MDCSIM, sets of services were identified using SIMs proposed in [4] and [37] considering the same business scenario described in Section 5. These approaches were chosen for comparison, because they are data-focused similarly to MDCSIM. Besides, among the methods presented in Section 2, they are the only ones that provide all the details necessary to their execution.

The first method [4] uses database reverse engineering to identify basic data access services. The practitioner chooses the tables considered relevant to the domain. Each table originates one data service. Then the practitioner chooses the CRUD operations comprised by this service within a set of CRUD operation signatures provided by the method. This method is classified as a Bottom-up approach, because it elicits only the IT perspective. The second SIM [37] identifies services using Data Flow Diagrams (DFDs). Every process in a DFD can be identified as a service. A coarse-grained process corresponds to composite services, while a fine-grained process, which cannot be decomposed, corresponds to an atomic service. This method is classified as a Top-down approach covering only the business perspective.

The use of MDCSIM and [37] was able to identify Entity services, Task services and Process services, supporting all the activities performed by the prioritized business areas. Nevertheless, the use of [4] was able to identify only Entity services, which do not cover all the activities executed within the business domain. Table 3 shows the correspondence of the services identified by using the three methods.

Table 3: Services identified by the three SIMs

| | MDCSIM | | [37] | [4] | |
|---|---|---|---|---|---|
| Type | Service | Operation | Service/Operation | Service | Operation |
| Entity | Cover | Create | | Cover | Create |
| | | Update | | | Update |
| | | Read | ReadCover | | Read |
| | | Delete | | | Delete |
| Entity | Premium | Create | | Premium | Create |
| | | Update | | | Update |
| | | Read | ReadPremium | | Read |
| | | Delete | | | Delete |

| | | | | | |
|---|---|---|---|---|---|
| Task | Premium_ Controller | register_Premium_ Payment | | | |
| | | register_Payment_ Delay | | | |
| Entity | Contractor | Create | | Contractor | Create |
| | | Update | UpdateContractor | | Update |
| | | Read | | | Read |
| | | Delete | | | Delete |
| Task | Contractor_ Controller | mark_For_revision | | | |
| | | suspend_ Contractor | | | |
| | | block_ Contractor | | | |
| | | register_ Bankruptcy | | | |
| | | unblock_ Contractor | | | |
| | | review_ Contractor | | | |
| Entity | Reinsured | Create | | Reinsured | Create |
| | | Update | | | Update |
| | | Read | ReadReinsured | | Read |
| | | Delete | | | Delete |
| Entity | Modality | Create | | Modality | Create |
| | | Update | | | Update |
| | | Read | ReadModality | | Read |
| | | Delete | | | Delete |
| Entity | Line | Create | | Line | Create |
| | | Update | | | Update |
| | | Read | ReadLine | | Read |
| | | Delete | | | Delete |
| Entity | Contract | Create | | Contract | Create |
| | | Update | | | Update |
| | | Read | ReadContract | | Read |
| | | Delete | | | Delete |
| Task | Contract_ Controller | expire_Contract | | | |
| Entity | Reinsurance | Create | | Reinsurance | Create |
| | | Update | | | Update |
| | | Read | ReadReinsurance | | Read |
| | | Delete | | | Delete |
| | | set_Risk_ Distribution | | | |
| | | update_Risk_ Distribution | | | |
| | | delete_Risk_ Distribution | | | |
| | | set_Contracted_Cover | | | |
| | | update_Contracted_Cover | | | |
| | | delete_Contracted_Cover | | | |
| Task | Reinsurance_ Controller | expire_Reinsurance | | | |
| | | aprove_ Reinsurance | | | |
| | | cancel_ Reinsurance | | | |

| | | | | | |
|---|---|---|---|---|---|
| Process | create_ Endorsement_ Process | create_Endorsement | | | |
| Entity | Risk_ Participant | Create | | | |
| | | Update | | | Update |
| | | Read | ReadRisk_ Participant | | Read |
| | | Delete | | | Delete |
| | | set_Retention | | | Delete |
| | | update_Retention | | | Delete |
| | | delete_Retention | | | Delete |
| Entity | Recovery_ Request | CreateRecovery_ Request | | Recovery_ Request | Create |
| | | Update | UpdateRecovery_ Request | | Update |
| | | Read | ReadRecovery_ Request | | Read |
| | | Delete | | | Delete |
| Task | Recovery_ Request_ Controller | send_RecoveryRequest | | | |
| | | approve_RecoveryRequest | | | |
| | | reprove_RecoveryRequest | | | |
| Entity | Claim_ Notification | Create | CreateClaim_ Notification | Claim_ Notification | Create |
| | | Update | UpdateClaim_ Notification | | Update |
| | | Read | ReadClaim_ Notification | | Read |
| | | Delete | | | Delete |
| | | set_Penality | Set_Penality | | |
| | | set_Regulator | Set_Regulator | | |
| Task | Claim_ Notification_ Controller | send_ClaimNotification | SendClaim_ Notification | | |
| | | reprove_ClaimNotification | ReproveClaim_ Notification | | |
| | | receive_ClaimNotification | ReceiveClaim_ Notification | | |
| | | send_to_Regulation | Send_to_Regulation | | |
| | | refuse_Claim | Refuse_Claim | | |
| | | close_with_ indemnity | close_with_ indemnity | | |
| Process | Analyse_ Claim_ Notification_ Process | Analyse_ClaimNotification | Analyse_Claim_ Notification | | |
| Entity | Claim_Entry | Create | CreateClaim_Entry | Claim_Entry | Create |
| | | Update | UpdateClaim_Entry | | Update |
| | | Read | ReadClaim_Entry | | Read |
| | | Delete | | | Delete |

| | | | | | |
|---|---|---|---|---|---|
| Task | Claim_Entry_ Controller | suspend_Claim_Entry | | | |
| Entity | Adjustment_ Request | Create | CreateAdjustment_ Request | Adjustment_ Request | Create |
| | | Update | UpdateAdjustment_ Request | | Update |
| | | Read | ReadAdjustment_ Request | | Read |
| | | Delete | | | Delete |
| Task | Adjustment_ Request_ Controller | approve_Adjustment_ Request | ApproveAdjustment_ Request | | |
| | | reprove_Adjustment_ Request | ReproveAdjustment_ Request | | |
| Entity | Payment | Create | CreatePayment | Payment | Create |
| | | Update | | | Update |
| | | Read | | | Read |
| | | Delete | | | Delete |
| Task | Payment_ Controller | confirm_Payment | | | |

Table 3 shows that MDCSIM offered the best coverage to the analyzed business domain. The services identified by MDCSIM have the same CRUD operations as the Entity services identified by [4] and also have operations that maintain attributes of classes related to the master data in the logical model (*set_Risk_Distribution, update_Risk_Distribution, delete_Risk_Distribution, set_Contracted_Cover, update_Contracted_Cover, delete_Contracted_Cover* from *Reinsurance* service and *set_Retention, update_Retention, delete_Retention* from *Risk_Participant* service). The services identified by MDCSIM also own operations that are not executed within the scope of the prioritized area, therefore are not explicit in the DFDs used in method [37], but are part of the master data lifecycle (*register_Premium_Payment, register_Payment_Delay* from *Premium_Cotroller* service; *mark_For_revision, suspend_Contractor, register_Bankruptcy, review_Contractor* from *Contractor_Controller* service; *expire_Contract* from *Contract_Controller* service; *expire_Reinsurance, aprove_Reinsurance, cancel_Reinsurance* from *Reinsurance_Controller* sevice; *create_Endorsement* from *Create_Endorsement_Process* service and *confirm_Payment* from *Payment_Controller* service).

The advantage of identifying the aforementioned operations is to provide service interfaces with wider scope, taking into account the ambit of the whole organization. In future iterations of the MDCSIM, the services that own these operations can be reused without the need to change their contracts (interfaces). This situation was demonstrated in the second iteration of MDCSIM, when the operations *suspend_Claim_Entry* of the *Claim_Entry_Controller* service, *close_with_Indemnity* and *refuse_claim* of the *Claim_Notification_Controller* service and *unblock_contractor* of the *Contractor_Controller* service, identified in the first iteration, were used for the Claim recovery process.

Another disadvantage of [37] in relation to MDCSIM is that, since [37] does not adopt a service layer approach, some Entity services can deal with both behavioral and attribute maintenance concerns. Such situation makes those services less stable as business process changes frequently.

The service portfolios identified by using each SIM were also compared by evaluating the reuse among iterations and the quality attributes of granularity, coupling, cohesion and entity convergence. These quality attributes were chosen because they are related with SOA goals of promoting business agility, increasing the ROI, and promoting business alignment. Business agility is related to the flexibility to respond to business changes. Coupling covers business agility goal. A service with low coupling is

more easily composed to support new requirements. Cohesion and entity convergence cover the ROI increasing, because services that encapsulate all actions of a business entity or are cohesive are more reusable, avoiding redundancy. Granularity covers the business alignment goal, because coarse-grained services offer rich functionality and have larger contribution to business processes. The evaluation of the aforementioned quality attributes was accomplished by using metrics proposed in [27]. The metrics are explained as follows.

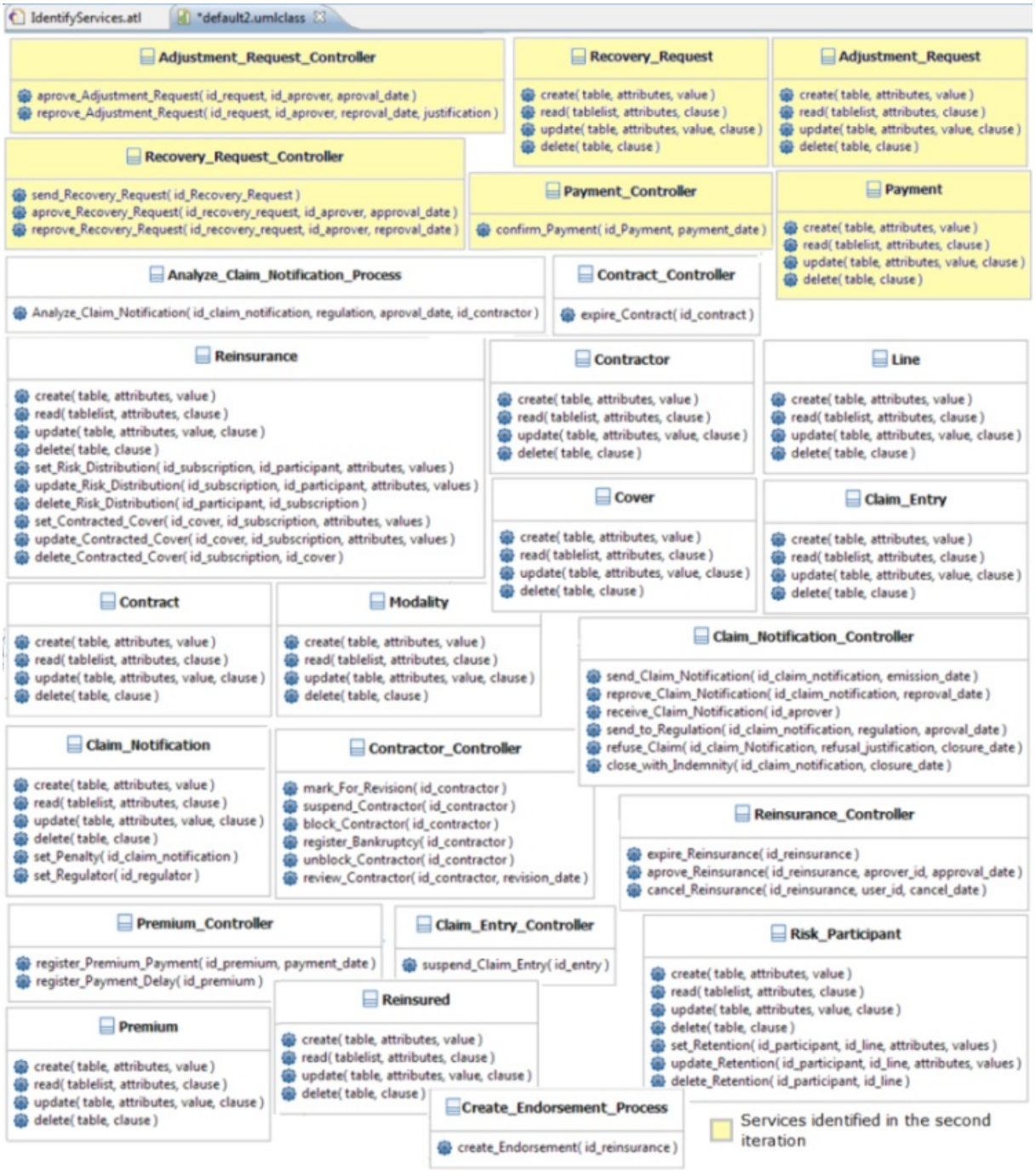


Figure 9: Class diagram of the complete list of identified services.

Table 4: Metrics value.

| | Granularity | Coupling | Cohesion | Entity convergence |
|---|---|---|---|---|
| MDCSIM | 3,76 | 307,00 | 0,76 | 1,56 |
| [37] | 1,00 | 478,00 | 1,00 | 2,20 |
| [4] | 4,00 | 134,00 | 0,57 | 2,00 |

**Service granularity** refers to the scope of functionality implemented by a service. Therefore, service granularity is defined as the average number of operations that the identified services have. Services should contain the highest number of operations as possible, defining a coarser interface that hides interaction details. **Service cohesion** refers to the degree of relation between the operations carried out by a service. Therefore, the degree of cohesion depends on the number of operations inside a service and on the dataflow between these operations. Cohesion should be the highest possible. **Service coupling** indicates the degree of interdependence between two services. An input of a service "A" can be obtained from an output of a service "B", thus resulting in the coupling of the service "A" with the service "B". As the requests to services are implemented through messages, the coupling degree is dependent of the number and the complexity of informational entities in the messages. Coupling should be preferably low. Business **entity convergence** is the extent to which a service focuses on processing operations of a specific business entity. Business entities provide a natural way to do a partitioning of business activities into services. A service encapsulating all the actions on a same business entity favors reuse because of its clear "business" functionality, thus entity convergence should be low [27].

The formulas to compute the aforementioned metrics can be found in [27]. To calculate the metrics, we assume that the relative complexities of the information entities are equal to the number of relationships their corresponding class has in the logical model. Complexity of Premium entity is 1, Cover entity complexity is 2, complexities of entities Contract, Contractor and Line are 3, complexities of entities Claim Entry and Adjustment request are 4, complexity of Modality entity is 5, complexities of entities Risk participant and Reinsured are 6, Recovery request entity complexity is 7, Payment entity complexity is 9 and complexities of entities Claim notification and Reinsurance are 12. For calculating service cohesion, we assume that the impact of data flows on the service cohesion is equal to 2 as in [27]. We also assume that the total number of activities is the total number of operations identified by each approach, as we are not working with process decomposition. Utility services were also not evaluated by the metrics, because the analyzed methods do not support their identification. The results of the metrics for each set of services identified by the three approaches are summarized in Table 4. The best values of granularity and coupling were obtained by [4]. Nevertheless, this method does not identify the Task services necessary to accomplish the Claim processing area activities, neither operations that maintain attributes of classes related to the master data in the logical model. The absence of Task services in this portfolio is also a reason for the low value of coupling, because this kind of service usually depends of the Entity services. Services identified in [37] had the maximum value of cohesion, because they have only one operation, thus granularity was low. The proliferation of fine-grained services is undesirable, since it increases service governance efforts. Coupling is very high because Task services depend on many Entity services as CRUD operations are scattered into several services.

The service portfolio identified by using MDCSIM has the best quality since it achieves a balance among the different metrics. The service portfolio generated by using MDCSIM had the best value of entity convergence and the second best values of granularity, coupling and cohesion. The best value of entity convergence was expected, because our method gathers operations accordingly to the master data manipulated by them. The value of cohesion metric shows that gathering operations by master data and by type of the operation (CRUD or operations that deal with entity behavior) contributed to provide

cohesive services. Good results in entity convergence and in granularity metrics suggest that our method is able to provide coarse-grained services that tend to be more reused since they (i) provide functionality of a business entity (providing a larger contribution to business processes) and (ii) avoid the multiple fine-grained interactions (several data flows are inside of services).

Analyzing service reuse between iterations, the employment of MDCSIM also had the best performance: 9 services identified in the first iteration (*Claim_Entry, Claim_Notification, Claim_Notification_Controller, Contractor_Controller, Line, Modality, Reinsurance, Reinsured* and *Risk_Participant*) were reused in the second iteration. Among the services identified by [4] only 6 services (*Claim_Entry, Claim_Notification, Line, Modality, Reinsured* and *Risk_Participant*) were reused in the second iteration. Finally, in [37] 6 services identified in the first iteration were reused in the second iteration (*CreateClaim_Entry, UpdateContractor, UpdateClaim_Notification, ReadClaim_Notification, ReadRisk_Participant* and *ReadModality*).

The aforementioned comparison is a simplified analysis that aims to provide an initial assessment on the capacities of MDCSIM to support SOA goals by promoting reuse among services and to identify services that deal with business and IT perspectives. It does not intend to be an experiment. We understand that this analysis has some weak points. Service identification was done only in one business scenario. Comparison should be repeated in other business scenarios for more evidences of the method validity. Another threat to the validity of our assessment is that all steps were executed by the same person, thus eliminating communication issues among the several roles involved in MDCSIM. The execution of MDCSIM should be repeated using different people for each role. Finally, it was not possible to compare the time and efforts to execute the SIMs, neither to assess the perception of MDCSIM utilization and learning. A controlled experiment should be done with groups of specialists and non-specialists in order to compare their performance in identifying services using MDCSIM and using other SIMs.

# 7 Conclusion

Different SIMs were proposed hitherto. However, few proposals consider both business and IT perspectives. In such proposals, there is a great dependency of detailed process description as inputs to elicit business perspective. Besides, the majority of the methods do not address the following research gaps in service identification: (i) identification of both business and software services, (ii) method configurability, (iii) candidate service quality assessment, (iv) elicitation of non-functional requirements and (v) analysis of economical aspects. In order to analyze both business and IT perspectives in service identification phase and also address the research gaps (i), (ii) and (iii), a method (MDCSIM) and a tool (MDCSIM plug-in) were developed.

MDCSIM is a Meet in the middle method that identifies services using as inputs master data and logical data models. Artifact-centric modeling technique is used within MDCSIM in order to identify master data lifecycle. MDCSIM plug-in is a MDA tool that automates the steps 3.4, 3.5 and 3.6 of MDCSIM. MDCSIM plug-in reads a UML class diagram (that represents master data attributes and relationships) and the UML state machine diagrams (that represent master data lifecycle) and identifies a list of Entity, Task and Process candidate services.

A proof of concept with MDCSIM was accomplished and comparisons with other two data-focused SIMs were performed. The proof of concept demonstrated the usage of master data and of the artifact-centric modeling technique to elicit business perspective (Claim processing area) and to identify business services (Task and Process services). The IT perspective was elicited by identifying master data attributes from databases used by the systems that support Claim processing area. A layer of IT services (Entity services) that abstracts data storage and ensures data integrity was also identified from the logical data models, therefore addressing the research gap (i).

The research gap (ii) was demonstrated in the proof of concept by slicing candidate services identification into 2 steps. First, only Entity services were identified for the business domain prioritized. Next, Task and Process services were identified in order to compose the complete set of services for the domain as presented in Figure 9. If an organization wants a "lightweight" version of MDCSIM, it can perform iterations in order to identify only Entity services (steps 3.1 to 3.4, and in the future execute steps 3.5 and 3.6 to identify Process and Task services. This feature is important for organizations with low maturity in SOA.

Research gap (iii) was addressed by using the metrics proposed by [27] to assess granularity, coupling, cohesion, and entity convergence service quality attributes. as presented in Section 6. This comparison suggests that MDCSIM identified services with more quality than [4] and [37], since it achieved a balance among the different metrics. The comparison also revealed that MDCSIM offered the best coverage to the analyzed business domain. MDCSIM was able to identify the operations necessary to fulfil business tasks and also identify operations that are not executed within the scope of the prioritized business area, but are part of the master data lifecycle. This characteristic affords services with wider interfaces than those identified by the other two data-focused SIMs, taking into account the scope of the whole organization. In subsequent iterations of the MDCSIM, the services that own these operations can be reused without the need to change their contracts, what is not accomplished by [4] and [37]. Finally, MDCSIM has had more services identified in the first iteration reused in the second iteration than the other two SIMs.

The aforementioned facts, suggests that MDCSIM can support the service identification phase in a service-oriented modeling and architectural design process. Nevertheless, future work can be done in order to address research gaps (iv) and (v) and also identify improvement opportunities by executing MDCSIM in other business scenarios.

## 8 Acknowledgements

This work was partly supported by the Brazilian funding agencies CNPq and FAPERJ.

## References

[1] S. Alahmari, D. D. Roure, and E. Zaluska. A model-driven architecture approach to the efficient identification of services on service-oriented enterprise architecture. In *Proc. of the 14th IEEE International Enterprise Distributed Object Computing Conference Workshops (EDOCW'10), Vitoria, Brazil*, pages 165–172. IEEE, October 2010.

[2] A. Arsanjani. Service-oriented modeling and architecture: How to identify, specify and realize services for your soa. Technical report, IBM, Software Group, 2004.

[3] A. Arsanjani, S. P. Ghosh, A. Allam, T. Abdollah, S. Gariapathy, and K. Holley. Soma: A method for developing service-oriented solutions. *IBM Systems Journal*, 47(3):377–396, July 2008.

[4] Y. Baghdadi. Reverse engineering relational databases to identify and specify basic web services with respect to service oriented computing. *Information Systems Frontiers*, 8(5):395–410, November 2006.

[5] M. Bell. *SOA Modeling Patterns for Service Oriented Discovery and Analysis*. Wiley, 2010.

[6] K. Bhattacharya, R. Hull, and J. Su. *A data-centric design methodology for business processes*, chapter 23. IGI Global, 2009.

[7] D. Birkmeier, S. Klockner, and S. Overhage. A survey of service identification approaches - classification framework,state of the art, and comparison. *Enterprise Modelling and Information Systems Architectures*, 4(2):20–36, December 2009.

[8] R. Boerner and M. Goeken. Service identification in soa governance literature review and implications for a new method. In *Proc. of the 3rd IEEE International Conference on Digital Ecosystems and Technologies (DEST'09), Istanbul, Turkey*, pages 588–593. IEEE, June 2009.

[9] R. B. Bradford. Efficient discovery of new information in large text databases. In *Proc. of the IEEE International Conference on Intelligence and Security Informatics (ISI'05), Atlanta, GA, USA, LNCS*, volume 3495, pages 374–380. Springer-Verlag, May 2005.

[10] S. Cai, Y. Liu, and X. Wang. A survey of service identification strategies. In *Proc. of the 2011 IEEE Asia-Pacific Services Computing Conference (APSCC'11), Jeju Island, South Korea*, pages 464–470. IEEE, December 2011.

[11] S. Chaari, F. Biennier, J. Favrel, and C. Benamar. Towards a service-oriented enterprise based on business components identification. In R. Goncalves, J. Muller, K. Mertins, and M. Zelm, editors, *Enterprise Interoperability II*, pages 495–506. Springer London, 2007.

[12] R. H. L. Chiang, T. M. Barron, and V. C. Storey. Reverse engineering of relational databases: Extraction of an eer model from a relational database. *Data & Knowledge Engineering*, 12(2):107–142, March 1994.

[13] M. J. Cho, H. R. Choi, H. S. Kim, S. G. Hong, Y. Keceli, and J. Y. Park. Service identification and modeling for service oriented architecture applications. In *Proc. of the 7th WSEAS International Conference on Software Engineering, Parallel and Distributed Systems (SEPADS'08), Cambridge, UK.*, pages 193–199. World Scientific and Engineering Academy and Society (WSEAS), February 2008.

[14] D. Cohn and R. Hull. Business artifacts: A data-centric approach to modeling business operations and processes. *IEEE Data Engineering Bulletin*, 32(3):3–9, September 2006.

[15] H. Demirkan, R. J. Kauffman, J. A. Vayghan, H.-G. Fill, D. Karagiannis, and P. P. Maglio. Service-oriented technology and management: Perspectives on research and practice for the coming decade. *Electronic Commerce Research and Applications*, 7(4):356–376, 2008.

[16] A. Dreibelbis, E. Hechler, I. Milman, M. Oberhofer, P. van Run, and D. Wolfson. *Enterprise Master Data Management: An SOA Approach to Managing Core Information*. IBM Press, 2008.

[17] T. Erl. *SOA: principles of service design*. Prentice Hall, 2007.

[18] A. Erradi, S. Anand, and N. Kulkarni. Soaf: An architectural framework for service definition and realization. In *Proc. of the 12th IEEE International Workshops on Enabling Technologies (WETICE'03), Chicago, USA*, pages 151–158. IEEE, September 2006.

[19] N. Fareghzadeh. Service identification approach to soa development. *International Journal of Computer, Information, Systems and Control Engineering*, 2(9):160–168, 2008.

[20] S. Gérard, C. Dumoulin, P. Tessier, and B. Selic. 19 papyrus: A uml2 tool for domain-specific language modeling. In *Proc. of the International Dagstuhl Workshop (DW'10), Dagstuhl Castle, Germany, LNCS*, volume 6100, pages 361–368. Springer-Verlag, November 2010.

[21] Q. Gu and P. Lago. Service identification methods: A systematic literature review. In *Proc. of the 3rd European Conference ServiceWave (ECS'10), Ghent, Belgium, LNCS*, volume 6481, pages 37–50. Springer-Verlag, December 2010.

[22] S. Inaganti and G. K. Behara. Service identification: Bpm and soa handshake. `http://www.bptrends.com/publicationfiles/THREE%2003-07-ART-BPMandSOAHandshake-InagantiBehara-Final.pdf`, last viewed January 2014, 2007.

[23] F. Jouault and I. Kurtev. Transforming models with atl. In *Proc. of the 2005 International Conference on Satellite Events at the MoDELS (MoDELS'05), Montego Bay, Jamaica, LNCS*, volume 3844, pages 128–138. Springer-Verlag, October 2006.

[24] T. Kohlborn, A. Korthaus, T. Chan, and M. Rosemann. Identification and analysis of business and software services - a consolidated approach. *IEEE Transactions Services Computing*, 2(1):50–64, January 2009.

[25] T. Kohlborn, A. Korthaus, T. Chan, and M. Rosemann. Service analysis - a critical assessment of the state of the art. In *Proc. of the 17th European Conference on Information Systems (ECIS'09), Verona, Italy*, pages 1583–1594, June 2009.

[26] D. Krafzig, K. Banke, , and D. Slama. *Enterprise SOA: Service-Oriented Architecture Best Practices (The Coad Series)*. Prentice Hall, 2004.

[27] Q. Ma, N. Zhou, Y. Zhu, and H. Wang. Evaluating service identification with design metrics on business pro-

cess decomposition. In *Proc. of the 2009 IEEE International Conference on Services Computing (SCC'09), Bangalore, India*, pages 160–167. IEEE, September 2009.

[28] A. Nigam and N. S. Caswell. Business artifacts an approach to operational specification. *IBM Systems Journal*, 42(3):428–445, July 2003.

[29] O. O. M. G. OMG. Mda guide v1.0.1. `http://www.omg.org/cgi-bin/doc?omg/03-06-01`, last viewed January 2014, 2003.

[30] S. Patig and H. Wesenberg. Role of process modeling in software service design. In *Proc. of the 7th International Joint Conference on Service-Oriented Computing (ICSOC-ServiceWave '09), Stockholm, Sweden, LNCS*, volume 5900, pages 420–428. Springer-Verlag, November 2009.

[31] K. Ponnalagu and N. C. Narendra. Deriving service variants from business process specifications. In *Proc. of the 1st Bangalore Annual Compute Conference (COMPUTE'08), Bangalore, India*, pages 4:1–4:9. ACM Press, January 2008.

[32] T. C. Shan and W. W. Hua. Service-oriented solution framework for internet banking. *International Journal of Web Services Research*, 3(1):29–48, January-March 2006.

[33] J. K. Strosnider, P. Nandi, S. B. Kumaran, S. P. Ghosh, and A. Arsnajani. Model-driven synthesis of soa solutions. *IBM Systems Journal*, 47(3):415–432, July 2008.

[34] T. Vale, G. B. Figueiredo, E. S. de Almeida, and S. R. de Lemos Meira. A study on service identification methods for software product lines. In *Proc. of the 16th International Software Product Line Conference - Volume 2 (SPLC'12), Salvador, Brazil*, pages 156–163. ACM Press, September 2012.

[35] S. Yongchareon, C. Liu, and X. Zhao. A framework for behavior-consistent specialization of artifact-centric business processes. In *Proc. of the 10th international conference on Business Process Management (BPM'12), Tallinn, Estonia, LNCS*, volume 7481, pages 285–301. Springer-Verlag, September 2012.

[36] A. T. Zadeh, M. Mukhtar, S. Sahran, and M. Khabbazi. A systematic input selection for service identification in smes. *Journal of Applied Sciences*, 12(12):1232–1244, 2012.

[37] Y. Zhao, S. Huayou, N. Yulin, and Q. Hengnian. A service-oriented analysis and design approach based on data flow diagram. In *Proc. of the 2009 IEEE International Conference on Computational Intelligence and Software Engineering (CiSE'09), Wuhan, China*, pages 1–5. IEEE, December 2009.

[38] O. Zimmermann, P. Krogdahl, and C. Gee. Elements of service oriented analysis and design. an interdisciplinary modeling approach for soa projects. `http://www-128.ibm.com/developerworks/webservices/library/ws-soad1`, last viewed January 2014, 2004.

## Author Biography

**Rosane S. Huergo** obtained her Master degree in Informatics at the Federal University of Rio de Janeiro (2014). Since 2003, Rosane Huergo has worked in projects of systems development, process modelling and optimization for several Brazilian companies. Currently, she is a System Analyst at Petróleo Brasileiro SA and is coordinating projects of IT governance, SOA and Enterprise Architecture.

**Paulo F. Pires** obtained his PhD in Systems Engineering and Computing at the Federal University of Rio de Janeiro (2002), with sandwich period at the University of Maryland. Currently, he is an Associate Professor at the Federal University of Rio de Janeiro and a Fellow of CNPq, productivity level 2 in technological development and innovative extension. Since 2002, Prof. Paulo Pires has coordinated several research and development projects, in partnership with companies such as EMC Research Center of Brazil, Embratel, Brazilian Navy, Secretary of Informatics of the State of Rio de Janeiro, Brazilian Ministry of Defense, and Brazilian Ministry of Aeronautics. Prof. Pires is the leader of the Research Group "UbiComp: Ubiquitous Computing, Adaptive Systems and Web Systems", a member of the research group ConSist (UFRN), and also integrates the Center for Distributed and High Performance Computing at the University of Sydney, where he develops research on complex distributed systems development.

**Flávia C. Delicato** received her PhD from Federal University of Rio de Janeiro in 2005. In 2010 she was a visiting academic in a post-doctoral stage at the University of Sydney, Australia. She is currently an Associate Professor of the Department of Computer Science at the Federal University of Rio de Janeiro, Brazil where she teaches for undergraduate and post-graduate courses, and integrates the Computer Networks and Distributed Systems Research Group and the Laboratory for Ubiquitous Computing (http://ubicomp.nce.ufrj.br/ubicomp/?lang=en). She is a Researcher Fellow of the National Council for Scientific and Technological Development (CNPq) and a Young Researcher Fellow from FAPERJ Brazilian Funding Agency. She has published more than 100 refereed international conference and journal papers. She is an Editorial Board Member of The Brazilian Journal of Computer Networks and Distributed Systems and of the International Journal of Computer Networks (IJCN). She has been participating in several research projects with funding from International and Brazilian government agencies, including CNPq (National Council for Scientific and Technological Development), CAPES (Coordenação de Aperfeiçoamento de Pessoal de Nível Superior), ANP (Agência Nacional do Petróleo, Gás Natural e Biocombustíveis ), RNP (Rede Nacional de Ensino e Pesquisa), Fundación Carolina (Spain), FAPERJ and AusAID (from the Australian Governemtn). She integrates the Centre for Distributed and High Performance Computing of the University of Sydney (http://sydney.edu.au/distributed_computing/). Her main research interests are: Wireless Sensor Networks, Ubiquituos Computing; Adaptive Systems; Middleware Model Driven Development; Web Service Technologies, WoT and IoT.