Decision Models as Software Artifacts
Bridging the Business-Software Gap for Decision Support Systems

Armando Guarnaschelli[1], Omar Chiotti[2] and Héctor E. Salomone[1]
[1]*INGAR, Instituto de Desarrollo y Diseño, CONICET-UTN, Avellaneda 3657, Santa Fe, Argentina*
[2]*CIDISI, R&D Information Systems Engineering Centre, FRSF-UTN, Lavaisse 610, Santa Fe, Argentina*

Keywords: Model Driven Development, Decision Models, Service Oriented Architecture, Reference Modeling.

Abstract: In this work we introduce a methodology for integrating the development of decision models with model driven software development approaches. This methodology captures the relationships between service oriented architecture software models and decision models by deriving from a common reference model, the data and conceptual elements of both types of models. Using this reference model and through successive model transformations and refinements, the methodology delivers integrated models and implementation models of software aimed to be more resilient to changes in business models. As all software artifacts are connected through a unique reference model, the collaboration with business partners is enabled at all levels by sharing or reusing existent reference models.

1 INTRODUCTION

Decision support community has produced along the years a rich body of highly efficient models and algorithms to solve relevant problems applicable to almost any aspect of decision making. The evolution of information systems towards complexity and integration of processes, challenges the survival of models and algorithms that were conceived and designed in isolation of the business context and software engineering considerations.

To unleash the full potential of model-based decision making in the context of modern software systems it is mandatory to create bridges among decision support specialists and software engineers. This integration can be facilitated by exploring the interactions of business modeling, service-oriented software development and Decision Problem (DP) analysis.

In this paper, we discuss a methodology that intents to make those interactions explicit and guide the development of business solutions including DPs all along its lifecycle. From the very early stages of requirement analysis and business process modeling to the final stages of deploying software integrated to complex software systems. The methodology is strongly based on service-oriented software development methods in recognition to the unpaired advantages this approach has to address the requirements of current business information systems.

From this well-established methodological backbone, we have made an effort to extend the main concepts behind SOA (Service Oriented Architecture, (Papazoglou & Van Den Heuvel, 2007) development to strength the interactions with business modeling and DP solving. A central idea is to maintain an always traceable relation among the software artifacts that are being generated and the business model. This is enabled by prescribing the specification of a unique reference model serving as the single source for domain representation, semantic interoperability and foundation of any involved data model.

Another important aspect is the adoption of a Model Driven Development (Hailpern & Tarr, 2006) approach that prescribes the generation of models at different stages of the development. These models provide the anchors for the automatic generation of the software artifacts required for the project by means of model to model transformations and eventually model to code transformations.

The methodology proposed in this work allows capturing, guiding and in some cases automating the reflection of changes at the business model into: Decision Models, their solution methods and strategies, as well as into Data models and software services enacting the processes in the business

Guarnaschelli A., Chiotti O. and Salomone H..
Decision Models as Software Artifacts - Bridging the Business-Software Gap for Decision Support Systems.
DOI: 10.5220/0004282000860095
In *Proceedings of the 2nd International Conference on Operations Research and Enterprise Systems* (ICORES-2013), pages 86-95
ISBN: 978-989-8565-40-2

model.

This characteristic is possible as this methodology establishes the specification of decision models and services independently of solution and implementation strategies. This contributes to bridging the business-software gap for decision support systems.

2 METHODOLOGY ROADMAP

The proposed methodology is schematically presented as a development roadmap in Figure 1 . It consists in four modeling workflows: Business Modeling, DP Modeling, Service Modeling and Reference Modeling.

The starting point is the Business Modeling workflow; its purpose is to produce a unified business model that will drive the development of every other workflow and becomes a permanent consulting source of requirements, processes, goals and objectives.

Reference Modeling is done throughout the lifetime of the project in order to capture relevant and essential domain elements used in every other workflow and data model. It resorts to an abstract and synthetic representation of the concepts in the business model including relationships between elements and possible constraints that validate instances of them with respect to the business context. Its purpose is twofold: First, it facilitates the interaction of the resulting SOA based software solution with existing systems, other enterprise systems and with decision model languages (such as GAMS (Brooke, Kendrick, Meeraus, & Raman, 2008), ILOG (IBM, 2011), etc.) and future systems to be developed.

Second it serves as the conceptual base of the project facilitating the communication between its stakeholders: decision support specialists, business analysts, potential users and software engineers.

In the DP Modeling workflow, the problem is identified and specified in the context of the business processes defined in the business model.

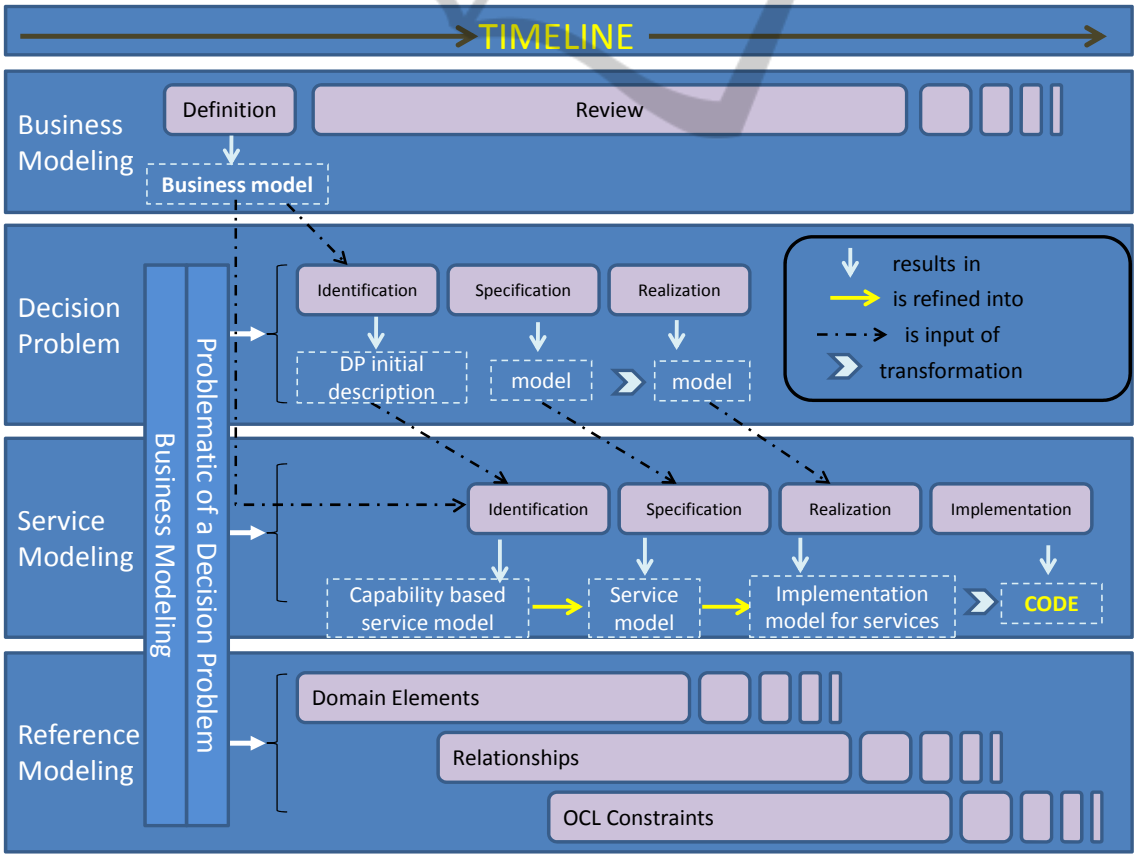

Figure 1: Methodology Roadmap.

This workflow considers both phases previous to solve the DP (the data, resources and actors involved) and post solving the DP (implementing and using the results). The purpose of this workflow is twofold: The first is obtaining a DP Specification independent of the solution strategy. This allows capturing every detail of the decision processes in the context of the business and also reflecting future changes in the business into decision processes and models. Second the workflow finishes with a solution strategy including the specification of a decision model in the context of this solution strategy (e.g. mathematical programming).

The Service Modeling workflow adapts approaches for SOA modeling in order to match the evolution of the DP Modeling workflow capturing the intermediate models as appropriate software artifacts.

DP and Service Modeling workflows have a common structure composed by the phases of: Identification, Specification, and Realization (resulting in an implementation model for both DP and services). The Implementation phase only applies for Service Modeling as it implies defining executable code for the SOA solution.

Several relationships are described in the roadmap, for instance: *results in* implies the activity/phase results into a software artifact (a model), *is refined into* implies the source model is refined into a richer model with increasing detail and introducing new elements, *is input of* implies that the source model is used to define the target model, *transformation* implies a model transformation into another model or code.

2.1 Business Modeling

Business modeling captures the vision and processes that an organization holds and intends to realize. The resulting models serve as the basis for the derivation of all software artifacts in a software project. Understanding the business means that software engineers, decision support specialists, and every stakeholder in a software project understand the organization structure, dynamics, goals, etc. A correctly built and specified business model provides a source of consultation on all business related issues during the software development lifecycle. Additionally if business modeling is done in conjunction with model driven development it provides the benefit of a resilient software solution, minimizing the effort to update software products given a change in the business model.

Business modeling is composed by the activities of business goals definition, business process modeling, business rules identification and requirements analysis. Following, these activities are briefly described:

The definition of business goals encompass the identification off all relevant goals of the business and their modeling as a hierarchical three of sub-goals and further refinement up to the definition of measurable objectives. These business goals and objectives must be related to one or more business processes for allowing the definition of metrics that measure the degree of goal fulfillment.

Business process modeling provides a detailed description of the business processes and sub-processes associated with a specific domain. The model includes all the business processes that are object of improvement and systematization and also includes other related processes that are needed to provide context and understanding of the business structure and dynamics.

There are diverse graphical modeling languages that can be used for business process modeling as Unified Modeling Language (UML) activity diagrams (OMG, 2010), Business Process Modeling Notation (BPMN) (OMG, 2011) and flowcharting among others. In this methodology we have chosen BPMN because it provides a common process description language that is both suitable for high-level, business-oriented description of the processes (as it is appealing for business analysts), and for low-level specifications (as required by software engineers).

2.2 Reference Modeling

The term reference model has been widely used in information system literature mostly to denominate any model which is used as a template to derive enterprise specific models for a given business area. A review of the usage of the term in the literature can be found in (Thomas, 2005), where it is characterized by its universality within a business area and its reusability in different system development projects.

In this software development methodology, the building of a reference model is considered a crucial step for integrating software designers and decision support specialists. Every model (software model, decision model, data model, etc.) will use a reference model with a specific significance and functionality as it serves to the following purposes:

• Semantic source: Every relevant concept related to the business model is included as an element in the reference model or can be defined by a valid combination of reference model elements. All meta-models in this methodology relate to the reference model by referencing a generic entity called *DomainElement*, which imply in general terms to any entity of the reference model.

• Foundation for the service data model. All messages exchanged among service providers and consumers are defined using reference model elements and the resulting data model constitutes the communication schema of the final SOA solution obtained. This schema is expressed using XML schema definition language (W3C, 2004).

• Basis to define the enterprise ontology when additional reasoning about the concepts of the reference model is required.

• Facilitating interoperability between business partners. Two business partners engaged in business collaboration will reduce their semantic interoperability problems by generating the reference model for the collaboration and addressing interoperability only at this level. This is possible due to the fact that a valid reference model must be able to represent every domain element used in the context of the business and as a consequence, every data element in any related system can be represented as a combination of reference model elements. Communicating two business related systems reduces to agree to use the same Reference Model.

To build a Reference Model UML class diagrams are proposed. Classes represent conceptual elements and their attributes help to specify a particular instance of a concept. Relationships between classes model relationships between concepts. Constraints that ensure valid reference model elements and instances should be expressed using OCL (OMG, 2010) which is a widely used semi-formal constraint language.

2.3 DP Modeling

As a guiding development principle for this phase, the identification and specification of the DP should be as rich as possible, and postpone the considerations about the solution strategies available and choice of reliable algorithms for the Realization phase. This strong separation between DP specification and how it is going to be solved reinforces the philosophy of this methodology which is to tighten the business solution (including the decision models) to the business model.

2.3.1 DP Identification

In organizational or inter-organizational business processes many decisions are taken either using an automated mechanism or by a manual tasks. In some cases the evaluation of the possible alternatives with respect to business goals and objectives can become both difficult and relevant enough to constitute a DP requiring a decision model for representation and solution. This methodology proposes to identify and put DPs in the context of the business model relating them to specific elements of the processes executed.

In general terms, any DP is composed by a set (finite or infinite) of alternatives (sometimes also called potential actions) and a set of one or more criteria (objectives) used to evaluate and compare alternatives (Figueira, Greco, Ehrogott, & Roy, 2005).

Within a business process a DP arises from two sources: *workflow decisions* and *decision tasks*.

A workflow decision determines the execution path to be followed in the process. If such decision needs to be evaluated against business goals and objectives and this evaluation is complex enough, the workflow decision determines a DP. Every alternative in this DP is an execution alternative for the business process. An example of a workflow decisions is the evaluation of a loan in a financial business process, whether the loan is granted or not different paths take place. If that evaluation is complex enough it may determine a loan grant DP.

Decision tasks are those that imply the achievement of a set of objectives with given characteristics or satisfying certain constraints and also may include the search for the right executors and the definition of a right course of actions. Every decision task intrinsically determines a DP. This work focuses only in those decision tasks requiring the formulation and solution of a decision model for their accomplishment. Examples of decision tasks are: Defining a production schedule, determining the optimal routes for a distribution company, and similar decisions.

In this phase of the DP workflow a DP is identified within a business process together with an initial description of alternatives and objectives derived from the business model as described in Figure 2. Every DP has a Problematic view referring to the way a decision model for the problem and feasible solutions should be built.

Problematic answers questions such: as in what terms should the problem be posed? What is the type of solution required?

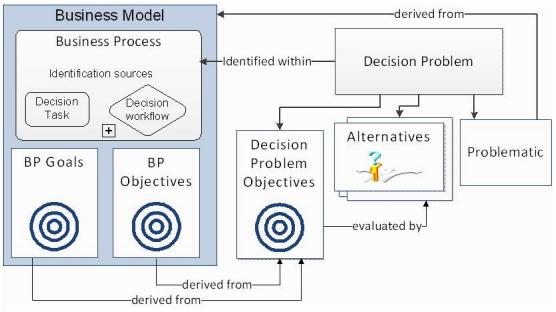

Figure 2: DP Identification.

How do the stakeholders see themselves with respect to the decision process? What kind of procedure seems the most appropriate for exploring the set of alternatives?

This methodology proposes to extend the problematic view to both DP identification and service identification workflows. Thus, the previous questions must be addressed in the context of the business process where the problem arises, including: who are the actors? And, what are the roles they play? How is the solution expected to be implemented in the business process and by which possible services people or processes.

Once a DP is identified and described, the next step obtaining a specification model. In the following section we propose a meta-model for that purpose.

2.3.2 DP Specification

In order to relate DPs to the business process they help to enact we have developed a meta-model consisting of a set of elements that allows specifying any DP in a high level of abstraction (Figure 3).

By adopting the specification meta-model, a DP specification results in a *Decision Model* that is composed by four main types of objects: *Variables*, *Parameters*, *Functional Relations* (between variables and parameters) and *Algorithms* that calculate functions required to describe the decision model.

This specification is intended to be independent of a solution strategy, and therefore the classes representing functional relations, variables and parameters should be instanced trying to represent the identified DP in a rich and expressive fashion without coupling to any solution paradigm

The concept of *Functional Relation* is extended by the concept of *Constraint* and *Objective Function* as these concepts are of common use when developing decision models. It is also extended to the concept of *Function* which might have an algebraic representation or require an algorithm for their calculation.

In a DP specification model, modeling and instance parameters can be used. *Modeling Parameters* are used to shape the structure of a model, e.g. a tolerance. *Instance Parameters* are the input data for the DP specification, e.g.: a forecasted demand in a production planning model. Instance parameters can be deterministic or non-deterministic (its variability should be considered).

Variables defined in this phase will not include specific variables related to a modeling technique. The same applies for the functional relations defined. For example, if the decision model requires constraints, the decision support specialists should not limit themselves to algebraic inequalities and equalities (the structure of a mathematical program), but should be able to define constraints such as:

$$\text{if(tasks } a \text{ and } b \text{ is assigned to resource } r\text{)}$$
$$\text{then } \{$$
$$[a \text{ precedes } b \leftrightarrow \text{endTime}(a) \leq \text{startTime}(b)\ ] \quad (1)$$
$$\text{or}[b \text{ precedes } a \leftrightarrow \text{endTime}(b) \leq \text{startTime}(a)]$$
$$\}$$

In constraint (1) the consequences of an assignment of tasks to a single resources is modeled. This example could be transformed in the following phase of the workflow DP Realization into a mathematical programming formulation with valid Big-M relaxations, or to a constraint programming formalism, etc. But strategically in this stage it is expressed in the natural way in which the constraint arises without coupling to the limitations of a solution technique.

It is relevant to analyze this example backwards, as relevant domain elements arise in its description, such as tasks having start and end times, resources, and the assignment relationship between tasks and resources. This clearly determines alternatives that belong to the previous identification phase, and *domain elements* that belong to the reference model.

The output of this phase will be a decision model uncoupled from a specific representation technique (mathematical programming, influence diagrams, simulation models for decision support, etc.) Many times a DP representation technique is both benefited and limited by a set of solution algorithms; this phase should capture a specification without these limitations. Without coupling does not mean the users are unable to use them, instead users should use all representation artifacts in their reach to better represent the DP.

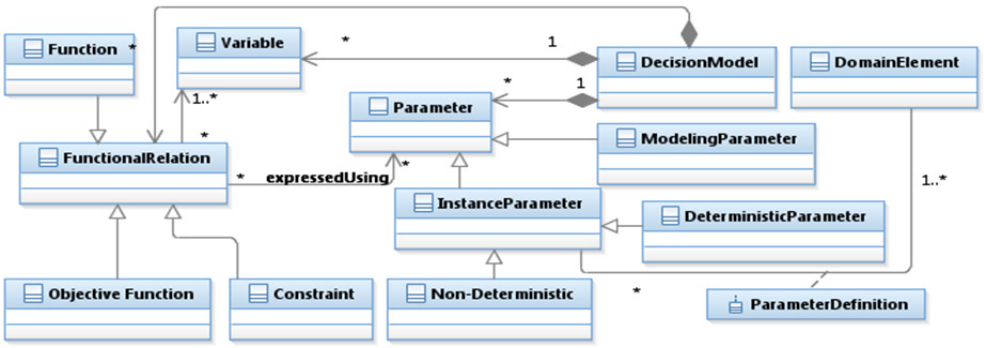

Figure 3: Methodology Roadmap.

The stronger the effort in characterizing the DP without coupling it to a specific representation, but relating it to alternatives and goals of a business model, the stronger the adherence to business requirements the business solution will have.

Traceability from the identified DP to the DP specification is required. It can be accomplished by exploiting the relationships in Figure 4. DP Alternatives express the alternative in terms of reference model elements (*Domain Elements*), and the relationship with variables in the specification model is captured by *Alternative Definition* objects that express the alternative in terms of reference model elements (domain elements) and hold a mapping with variables in the specification model. This allows a refinement on the representation of alternatives, because usually alternatives cannot be mapped directly to variables. Moreover, to represent an alternative, many variables may be required. For example a route in a logistic problem may be represented by a set of nodes and for each node a variable is required, therefore for this alternative (the route) a set of variables is required.

2.3.3 Decision Problem Realization

The design of a solution to the specified decision model representing a DP is captured in a DP Realization model.

In order to obtain such model, decision support specialists will explore the technical feasibility of different techniques to formulate and solve decision problems. It is in this realization phase where decision support specialists need to evaluate different algorithms and modeling languages and solver engines available for the solution strategy defined.

According to the DP Specification model, that is, the type of functional relations, variables, objective functions and constraints, the correct technique or combinations of techniques will be chosen.

The tasks involved in this realization phase encompass the tasks defined in the phases of design and choice in the Simon´s approach to rational decision making (Simon, 1977). In the design phase, alternative solutions to the problem are developed and explored and in the choice phase, a course of action is selected. The requirement for this phase is to obtain a solution that satisfies a realization contract between the DP specification and realization models as shown in Figure 5. The requirements for this contract are of two types, contractual and operational.

Contractually a realization model has to be able to fulfill as good as possible with the specification of the DP considering its variables, functional relationships and objectives. During the formulation of a realization model the problematic aspects of the DP should be revised as they affect the feasibility of implementing the obtained solution in the context of the business processes.

The realization contract has to establish a well-defined link with the specified DP. Decision support specialists should document how specified variables, constraints and objectives are maped to the realization model. Every aspect of the realization model referring to business related concepts should be included in the specification model before its usage in the realization model. The realization model may introduce its own related concepts as needed by the solution strategy but they should be clearly differentiated from the former ones.

Operationally in the specification phase of the DP and services (depicted in the following sections) it is established that parameters and the solution of the DP and messages of the service model should be expressed using reference model elements. For that purpose an automatic transformation engine that translates the DP specified parameters and solution to the realization model parameters and solution

should be built. From the point of view of decision support specialists it means that the parameters needed to solve the realization model should be built using combinations and transformations of Reference Model elements.

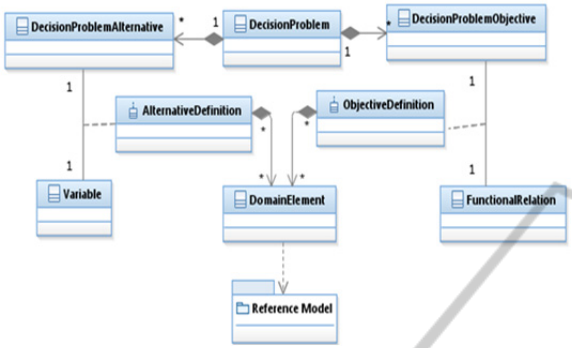

Figure 4: Traceability of a DP from Identification to Specification.

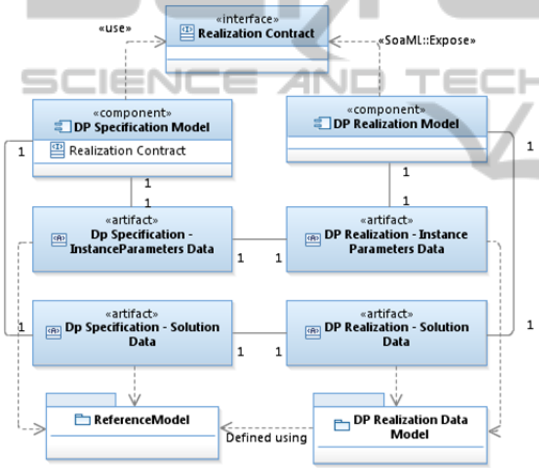

Figure 5: DP Realization Contract.

The automatic transformation engine will realize the one to one relationships between the data artifacts of both specification model (as required to be used by the business model) and realization model (as required to for the realization model to be solved).

2.4 Services Modeling

The main output of this phase is a model that specifies all the potential services needed to enact the business process in coordination with the decision model identified and specified previously. To define this model we resort to the SOAML (OMG, 2009) modeling language, in which *Capability* is the main concept used to identify potential services, and serves as container for *Operations* that can be seen as related functions.

For this phase, SOAML supports many identification and development techniques; however this work proposes an explicit link between every capability and some aspect of the business model or the decision model (DP specification model) that can be labeled as the "source" for that capability. Therefore the concept *Capability Identification Source* is introduced. This source can be either a specific BPMN element of the business process or elements of the DP specification model. The introduction of this dependency has the purpose of ensuring future developed services can be traced back to either a function needed by the execution of a business process or the solution of a DP. Capabilities also may be related to one specific participant in a collaborative process, in Figure 6 represented by a *pool* in such case the capability will hold operations able to execute every task and activity (sub-process) the participant performs in the business process. A capability can also be related to one specific role/resource, in Figure 6 represented by a *Lane*, in such case the capability belongs to a participant in the business process but defines the operations related to the role the participant assumes in a section of the business process. For example a planner participant in a production system may take the role of forecasting demand and the role of creating a production schedule, and SOA designers may choose to provide different capabilities for both functionalities. These aspects are shown in the Service Identification Meta-model shown in Figure 6. On the left of the figure Business process related capabilities are identified and on the right DP related capabilities.

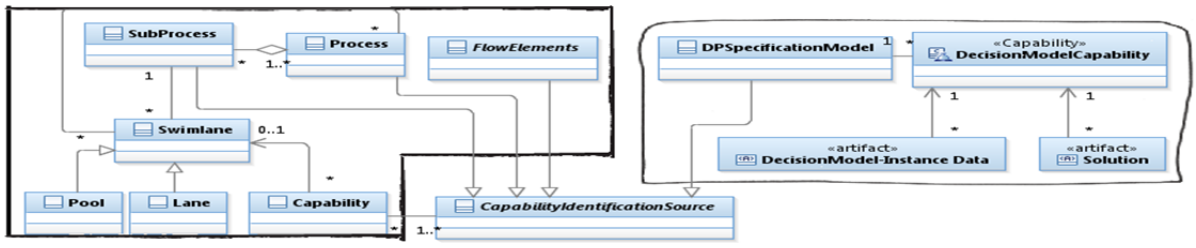

Figure 6: Service Identification Meta-model.

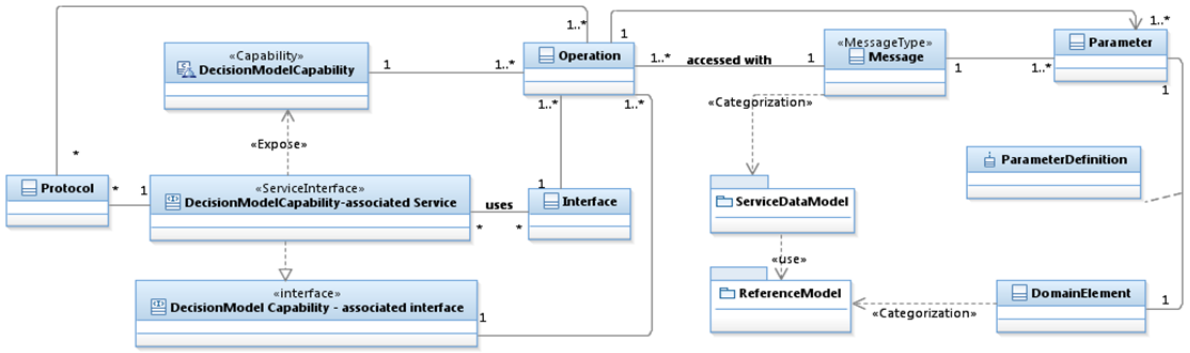

Figure 7: Service Specification Meta-model.

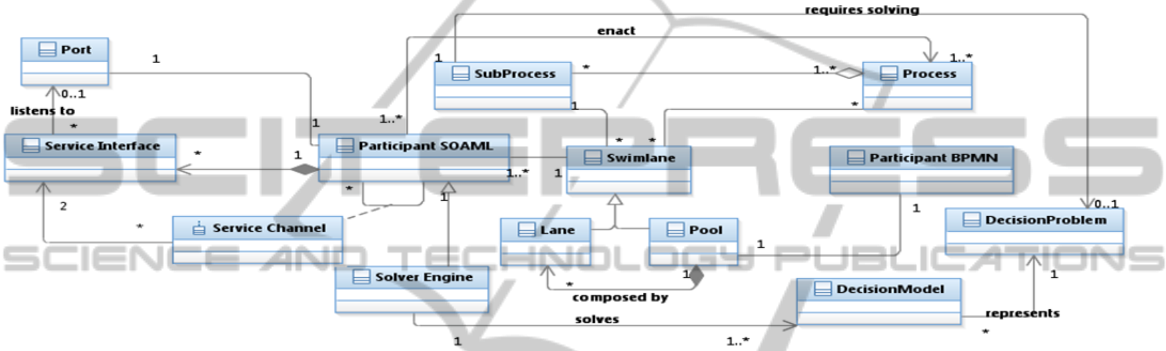

Figure 8: Service Realization Meta-model.

In this phase the problematic aspects identified in DP Identification must be reflected on the choice of the capabilities and operations needed to realize the business processes in the business model. The result of this phase is a typed initial service model represented by Business Process and DP related capabilities and a possible initial services architecture.

2.4.1 Service Specification

SOAML provides all the needed conceptual elements to specify a service. This is done by defining service interfaces (including operations, protocols, usage contracts), their realizations (exposed interfaces), and the request of other services, depicted as usage relationships.

Every service operation requires parameters for its execution. Instead of using a set of parameters for accessing an operation, this methodology proposes a document centric approach, in which each operation is associated with one single message with a document attached containing all the required data defined in compliance and derived from the reference model.

In Figure 7 these concepts are organized into the Service Specification meta-model. Although the concepts in this figure apply to any capability in the SOA solution, we only depict those capabilities related to the decision model functions.

Every message is defined within a *ServiceDataModel* package, which contains all messages in a service solution. This package is defined using the reference model. A service data model establishes the subset of *ReferenceModel* domain elements, and their relationships, required to be able to define every message in the service solution.

Protocols define the interaction between the service and its consumers. Protocols can be defined using UML interaction diagrams, that later can be transformed into a service choreography (artifact to define a collaboration between different services) or into a service orchestration (artifact to define internal interaction of a compound service).

An emphasis should be made in order to make an explicit relationship between the DP specification model and the service model. Parameters in a DP (Figure 3) are captured in the service specification model as well but they are hold by one or more messages of the *ServiceDataModel*. Every parameter of the specified decision model must have a parameter definition built up with reference model elements.

2.4.2 Service Realization

In this phase software developers need to take decisions to answer questions such as: which service providers will provide which services? How services operations will be implemented determining the internal behavior of a service? How services providers will be assembled to constitute participants in the business process enactment by services? How service channels, to provide communication between participants will be realized?

After answering all these questions the SOA solution is ready for its implementation. Many authors have dealt on how to anwer these questions properly (Bercovici, Fournier, & Wecker, 2008; Flaxer & Anil, 2004).

In Figure 8 we provide a meta-model for the realization and composition of services regarding to the objectives of this methodology. We propose that SOA participants be always related to one *Swimlane* which could be either a lane (a role of a participant in BPMN business process) or a pool (a participant in a BPMN business process). Each SOA *Participant* will assemble a set of services and will provide the corresponding ports to listen the request for its services. In this way, the business process will be realized by a set of SOAML participants (service providers) through their interactions. The flow of messages in the business process is realized by the exchange of messages in the service interfaces exposed by the participants. There is not a one to one relationship between a SOAML participants and business process participants because to provide the functionality of a BPMN participant one or more SOAML participants can be required. The abstraction provided by the concept of participant, allows the enactment of the business process regardless of the actual implementation of the service, which can in practice be deployed across multiple enterprises, via web services, or within a given single enterprise service bus (the communication channel within an enterprise).

In particular reference to the implementation of the service for supporting the realization strategy for the decision model, Figure 8 shows the *Solver Engine* as a participant offering the service to solve the decision model.

3 VALIDATION

The methodology presented here was applied and validated in the development of a software system for collaborative management of disruptive events in supply chains. This project, used as a case study, included the definition of a complete new business process conceived to be executed collaboratively by independent supply chain partners with the intention of providing system support for companies willing to engage in collaboration agreements for controlling the execution of their supply processes.

As result of the business model analysis, the collaborative management of disruptive events in a supply chain was modeled as a collaborative business process that specifies a set of decision making activities that require complex models to systematize the capture of information about internal and external changes, the prediction of disruptive events that can affect the schedule execution, and the activities of feasibility checking and schedule repair considering the distributed nature of a supply chain.

The resulting business solution consisted in a service-oriented information system where standard SOAML techniques were applied following the guidelines of the methodology of this work in order to derive the architecture, define the services interfaces, the service data models and the choreographies representing the collaborations.

The proposed reference model accomplished the description of the problem information in a very high level of abstraction and therefore is applicable to a wide range of supply chain processes, from procurement, manufacturing, distribution, and retailing domains.

In particular, the reference model proposed has the characteristic of providing self-contained descriptions of the information required for the decision making activities involved in the business process. This feature enables the possibility of automating the generation of decision models expressed in standard representations for decision making tools (as mathematical programming solvers or inference engines)

The details of the case study describing the business process, the reference model and other artifacts produced as the result of each phase can be found in (Guarnaschelli, Fernandez, Chiotti, & Salomone, 2012).

4 CONCLUSIONS

The methodology proposes to identify a DP starting from the context of an enterprise business model, specifically where it arises, in the enterprise business processes. Current DP and modeling practices tend to formulate decision models to solve DPs without

prescribing an explicit link between model formulation and the enterprise business model. It also formally preserves these relationships for tracing purposes all along the development life cycle. By ensuring a well-defined link between the business model and the solved DP, more reliable and usable Decision Support Systems can be obtained. This is possible because the link obligates to consider the context of the decision which is composed by people, resources, data sources, workflows, goals, business rules, etc. The risk of not making this explicit link is that even a good solution for the DP fails at its insertion in the business.

In current practice, the model formulation is generally biased by the modeling and solution paradigm chosen for the formulation (e.g. mathematical programming models). The adoption of a particular paradigm normally forces to make some strong hypothesis about the DP (e.g. avoiding nonlinear constraints, hypothesis on unknown but required probability distributions, among others). As a result, a premature abstraction of the DP normally takes place with the risk that the models obtained do not represent correctly the actual problem by accepting limitations or formulating assumptions without enough business information. In the proposed methodology, the identification and specification of the DPs are conducted independently from a solution or formulation paradigm using a general meta-model.

Sometimes the lack of a unified data sources and models results in the introduction of concepts and entities for the DP formulation and solution without connection to the business process, where messages are exchanged between participants and software systems that might not have a close and easily to capture relationship with DP concepts and entities. As a consequence a misalignment between the required data to solve the DP and the existent data exchanged in messages in the business processes arises, causing implementation and integration problems both at the syntactic and semantic level. The methodology proposes that once a solution paradigm is finally adopted, its formulation relies always on combining existent elements in the reference model, avoiding the aforementioned misalignment.

ACKNOWLEDGEMENTS

This work is partially supported by Consejo Nacional de Investigaciones Científicas y Técnicas (CONICET) and Agencia Nacional de Promoción Científica y Técnica (ANPCyT).

REFERENCES

Bercovici, A., Fournier, F., & Wecker, A. (2008). From Business Architecture to SOA Realization Using MDD. In I. Schieferdecker & A. Hartman (Eds.), *Model Driven Architecture – Foundations and Applications* (Vol. 5095, pp. 381-392): Springer Berlin / Heidelberg.

Brooke, A., Kendrick, D., Meeraus, A., & Raman, R. (2008). *GAMS User Guide*

Figueira, J., Greco, S., Ehrogott, M., & Roy, B. (2005). Paradigms and Challenges *Multiple Criteria Decision Analysis: State of the Art Surveys* (Vol. 78, pp. 3-24): Springer New York.

Flaxer, D., & Anil, N. (2004, 15 Sept. 2004). *Realizing business components, business operations and business services.* Paper presented at the E-Commerce Technology for Dynamic E-Business, 2004. .

Guarnaschelli, A., Fernandez, E., Chiotti, O., & Salomone, H. E. (2012). A service-oriented approach to collaborative management of disruptive events in supply chains. *International Journal of Innovative Computing, Information and Control, 8*(7 B), 5341-5368.

Hailpern, B., & Tarr, P. (2006). Model-driven development: the good, the bad, and the ugly. *IBM Syst. J., 45*(3), 451-461.

IBM. (2011). IBM ILOG CPLEX Optimization Studio. from http://www-01.ibm.com/software/integration/optimization/cplex-optimization-studio/

OMG. (2009). Object Management Group - SoaML Version 1.0 - Beta 2. from http://www.omg.org/spec/SoaML/1.0/Beta2/PDF

OMG. (2010). Object Management Group, Unified Modeling Language (UML) 2.3. from http://www.omg.org/spec/UML/2.3

OMG. (2011). Object Management Group , Business Process Model and Notation (BPMN) 2.0 from http://www.omg.org/spec/BPMN/Current

Papazoglou, M. P., & Van Den Heuvel, W. J. (2007). Service oriented architectures: Approaches, technologies and research issues. *VLDB Journal, 16*(3), 389-415.

Simon, H. A. (1977). *The New Science of Management Decision*: Prentice Hall PTR.

Thomas, O. (2005, September 5, 2005). *Understanding the Term Reference Model in Information Systems Research.* Paper presented at the Business Process Management Workshops, Nancy, France.

W3C. (2004). XML Schema Part 0: Primer Second Edition. from http://www.w3.org/TR/2004/REC-xmlschema-0-20041028/