A Semantic-Based Framework for Discovering Business Process Patterns

Laden Aldin, Sergio de Cesare and Mark Lycett
Department of Information Systems and Computing Brunel University
Uxbridge, United Kingdom

Abstract. Patterns currently play an important role in modern information systems (IS) development and their use has mainly been restricted to the design and implementation phases of the development lifecycle. Given the increasing significance of business modeling in IS development, patterns have the potential of providing a viable solution for promoting reusability of recurrent generalized models in the very early stages of development. This paper focuses on business process patterns and proposes an initial framework for the discovery and reuse of business process patterns within the IS development lifecycle. The framework synthesizes the idea from the domain engineering literature and proposes the use of semantics to drive both the discovery of patterns as well as their reuse.

Keywords: Pattern, Information system development, Business process pattern, Business processes, Domain engineering.

1 Introduction

Business modeling is assuming increasing significance in information systems (IS) development. Evidence of this phenomenon is highlighted, for example, by the introduction of a business modeling phase in methodologies like the Rational Unified Process, the recent definition of the Business Process Modeling Notation (BPMN) and the emergence of service-oriented approaches in which services are combined to realize business processes. Despite these positive signs modeling business processes remains problematic due to the evolutionary nature of organizations. Business processes evolve throughout an organization's lifetime in order to meet dynamic and changing business requirements [1]. It is essential that such changes are represented systematically and their impact is clearly understood. When developing computer-based information systems, it is necessary to understand the role they play in giving support to their business context. To reach such understanding there is a need to create business process models [2]. Business process modeling (BPM) is frequently used to control the execution of organizational processes and to ensure consistency and thoroughness in capturing relevant processes to improve efficiency and productivity. The achievement of greater agility and flexibility within BPM represents a key goal for organizations. One of the reasons that impede BPM to achieve this goal is the lack of systematic reuse of business models. In IS development business modelers may encounter similar and recurrent patterns of behavior. Being able to

reuse previously modeled behavior can have a beneficial impact on the quality and efficiency of the overall IS development process and also improve the effectiveness of an organization's business processes [3].

The representation of organizational processes has been the focus of much research in past years. Only some of it has focused on modeling business-related patterns [4]. This paper provides a contribution in this sense. More specifically, this study focuses on business process patterns. Business process pattern is a reusable model to the solution of a particular problem. It offers a solution based on previous success in resolving a similar type of business problem.

The remainder of this paper is structured as follows: the following section provides an overview of the background related to patterns in IS development and business process modeling. Section 3 presents a semantic-based framework for the identification of business process patterns as well as their reuse. Section 4 presents a worked example demonstrating the application of the framework. Finally, section 5 presents conclusions and an outline of future.

2 Background

The concept of patterns was introduced by the architect Christopher Alexander in 1977. Alexander et al. [5] refer to patterns in the following way: "Each pattern describes a problem which occurs over and over again in our environment, and then describes the core of the solution to that problem, in such a way that you can use this solution a million times over, without ever doing it the same way twice". Beck and Cunningham [6] initially introduced patterns in software programming by adopting ideas and principles first described by Alexander [5] in the field of civil architecture. The pattern concept was developed further and introduced at a design level. Examples of initial design patterns modeled by Coad [7] included 'item description', 'time association' and 'event logging'. Coad et al. [8] later adopted the term archetype to indicate "a form from which all classes of the same kind more or less follow" (p.3). Design patterns finally became a mainstream architectural technique thanks to Gamma et al. [9] who systematically compiled a catalogue of over 20 design patterns.

Subsequently patterns were introduced by Hay [10] to represent generic data structures typically used to model the information requirements of business organizations. Similarly to Hay, Fowler [11] defined a set of analysis patterns with the intention of reflecting "conceptual structures of business processes rather than actual software implementations" (p.xv). The works of both Hay and Fowler mainly focused on structural patterns (data/information). Some process patterns can be identified in Fowler, but these remain mainly underdeveloped. Furthermore Fowler's work tends to be directed toward software designers. As a result his analysis patterns in many areas commit more to software artifacts rather than to generic business domain structures and behavior.

Eriksson and Penker [3] later developed a set of business patterns, which came closer to a generic representation of organizational structures and processes. Although these patterns like the previous (Fowler and Hay) are ultimately aimed toward the facilitation of realizing software artifacts that will help to effectively and efficiently

develop and 'run' information systems, Eriksson and Penker's business patterns are modeled and described from a perspective that is closer to that of the enterprise rather than the software developer.

More recently there has been an increased interest in business process patterns specifically in the form of workflows. This greater interest is primarily due to the emergence of the service-oriented paradigm in which workflows are composed by orchestrating or choreographing web services. van der Aalst et al. [12] produced a set of so called workflow patterns. This initiative started by systematically evaluating features of workflow management systems and assessing the suitability of their underlying workflow languages. However, as Thom et al. [13] justly point out, these workflow patterns are relevant toward the implementation of workflow management systems rather than identifying business activities that a modeler can consider repeatedly in different process models. In fact the workflow patterns of van der Aalst et al. [12][14] are patterns of reusable control structures (for example, sequence, choice and parallelism) rather than patterns of reusable business processes subject to automation. As such these patterns do not resolve the problems of domain reuse in modeling organizational processes.

Besides the debatable business nature of the patterns discussed above, a more important limitation can be identified. In the patterns literature the way in which patterns are discovered is not clear. The literature states that patterns derive from experience and that a model constitutes a pattern if it has been used in multiple instances to resolve the same type of problem. Within the business domain, knowledge and experience tends to be dispersed among diverse and numerous sources (e.g., people, documents, legacy applications, designs and data, etc.). Often such knowledge is implicit and/or even informal and business behavior is not just designed, but is in good part emergent.

With more and more researchers and practitioners recognizing the importance of reusability in business process modeling [15], it is essential to explore new viable solutions that can provide successful ways to reuse. This paper proposes the adoption of semantics in order to discover new business process patterns and subsequently apply such patterns when modeling businesses. This study aims at overcoming two problems with previous solutions: (1) as highlighted above, limited work has been carried out on by other authors on business processes patterns, and (2) none of the previous work provides guidelines to modelers as to how business process patterns can be discovered. The following section proposes a semantic-based framework that can help overcome such problems.

3 SDR Framework

This paper proposes a framework for the semantic discovery and reuse of business process patterns. Patterns are initially discovered from legacy sources and then applied during business modeling. The framework is based on a dual lifecycle model as proposed by the domain engineering literature [16]. This model defines two interrelated lifecycles (Figure 1): (1) a lifecycle aimed at generating business process patterns and (2) a lifecycle aimed at producing business process models. To model an

organization in terms of its information rather than simply the data flowing through it requires understanding of the meaning of that information, its semantics. Semantics play a key role in this framework and are modeled through ontologies. While ontologies are used to represent the process patterns in the former lifecycle, the patterns' semantics then drive subsequent business modeling efforts during the latter lifecycle.

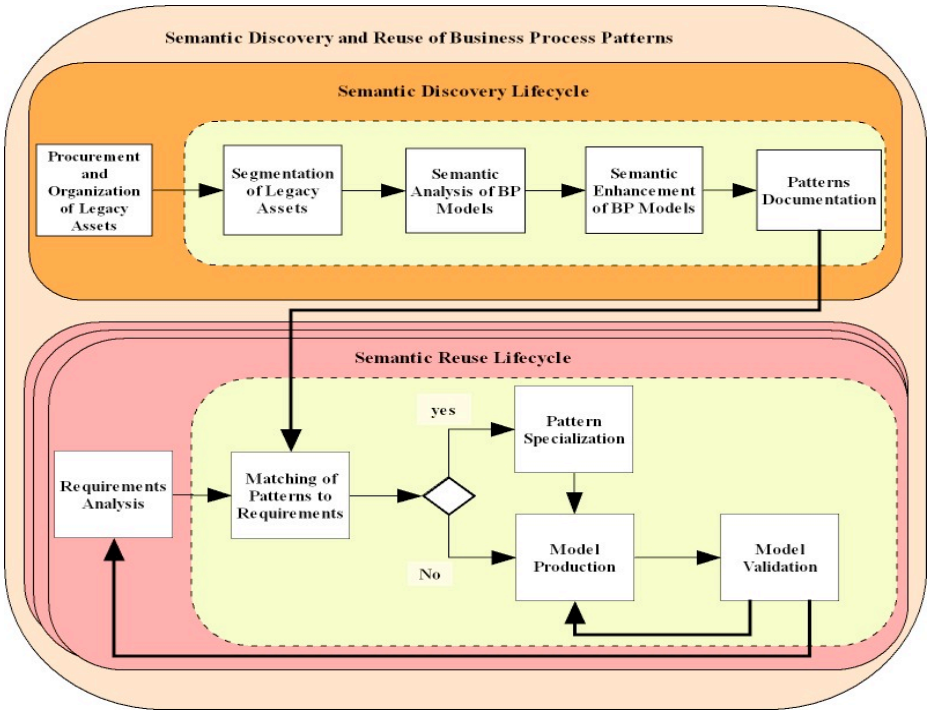

Fig. 1. SDR Framework (adapted from [17])

Theoretically speaking a semantics-based approach to modeling must ensure that there is evidence of mapping between elements of a model and the real-world things that those modeling elements refer to. This concept of mapping is integral to most definitions of semantics whereby there is a relation between a signifier (sign or symbol) and the signified (the thing being represented). Evidence of such mapping within the proposed framework derives from legacy source data. In this study legacy sources represent any body of knowledge (system application data, documentation, models, expert knowledge, observations, etc.), which provides confirmation of the existence of certain behavior and types of behavior in an organization. For example, from organizational documentation of a bank a modeler may elicit behavior corresponding to the withdrawal of money from an account. This behavior can be detailed into a series of steps that lead to a certain outcome (e.g., an account being debited).

From an ontological perspective it is necessary to answer a couple of fundamental questions when acknowledging the existence of something. Firstly, what does it mean for something to exist? Secondly, given a thing (anything) what is it? The first question implies making some choices as to how we view the world, i.e. the modeling paradigm adopted. The second question implies having to determine what something represents within the chosen paradigm. For example, if the entity-relationship paradigm were chosen then a specific withdrawal could be represented by an entity and the specific amount withdrawn could be represented by an attribute. In a process-based paradigm (e.g., the Business Process Modeling Notation (BPMN)), a withdrawal could correspond to a process and an account could correspond to a data object.

The paradigm that the proposed framework is based on is the object paradigm as proposed by Partridge [18]. The object paradigm (not to be confused with the object-oriented paradigm) determines the existence of an object (or thing) through its spatio-temporal extension. Hence, in the object paradigm a thing exists because it has a spatio-temporal extension in our universe. This statement provides an answer to the first question formulated above. In terms of the second question, Figure 2 provides a broad overview of the types of objects that the object paradigm acknowledges the existence of. These include:

- Individuals: Things with a four-dimensional (4D) extension and, unlike classes, do not have instances (hence the name individual);
- Classes: Types or sets of similar objects. The extension of a class is given by the extensions of all its instances;
- Tuples: Relationships between objects;
- Tuple Classes. Classes (or types) of tuples;
- Temporal parts: Temporal parts of 4D objects. They are specialized into states and events.
- States: Temporal parts with duration. A special predecessor relationship can exist between states whereby one state temporally precedes another;
- Events: Temporal parts that occur instantaneously thus having no duration. Particular classes of events are creations and dissolutions. The former represent events from which objects are generated, while the latter represent events that dissolve or terminate an object. Events happen at specific time instants and happen to one or more objects.

The object paradigm is, in terms of its use of four-dimensional objects, consistent with the perdurantist philosophical theory of persistence and identity [19]. It is beyond the scope of this paper to delve into the minute aspects of the paradigm. For detailed explanations on the object paradigm the reader is referred to Partridge [18].

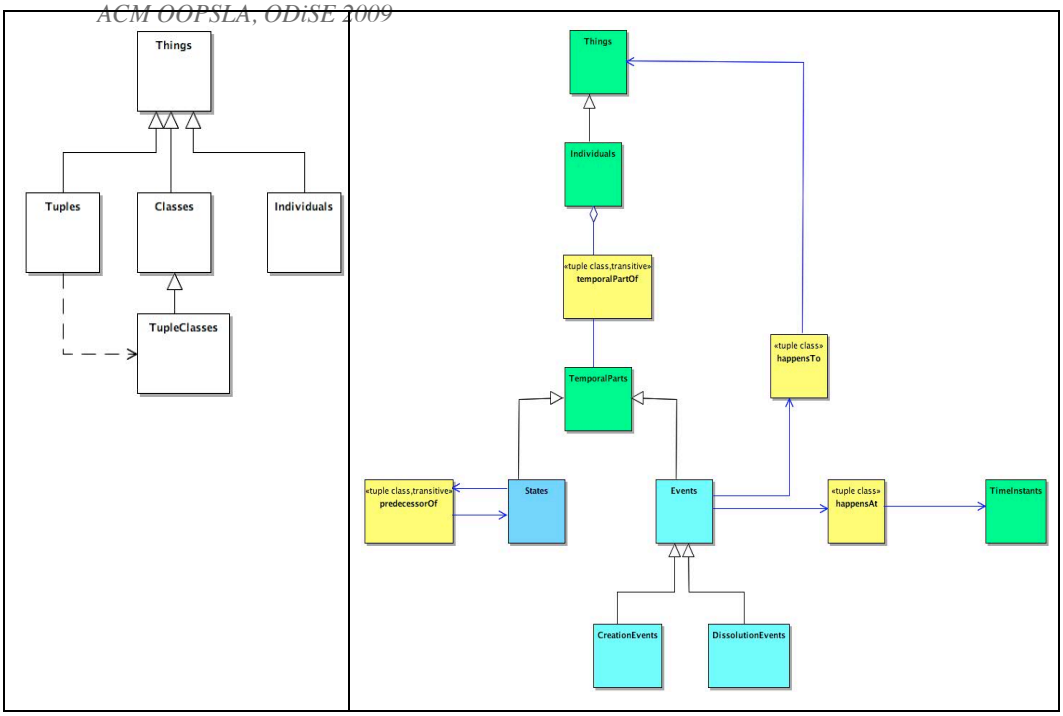

Fig. 2. Foundation of the Object Paradigm

3.1 Ontology of a Business Process

Business process is normally defined as a set of interrelated activities aimed at achieving one or more organizational goals. Understanding the ontological nature of a business process, however, requires more than just a definition. Definitions can be ambiguous. For example, what is an activity and how is an activity different from a process? Moreover, when can a set of activities be said to achieve one or more goals? Determining the ontology of a business process means providing an answer to the question: what is a business process in the chosen paradigm of representation?

When an organization carries out what is normally known as a business process, the organization and/or one of its parts (i.e., organizational units) enters a particular state; for example, the state of 'withdrawing $300 from John Smith's current account'. This state is initiated by an event, which in the particular example is triggered by the client 'John Smith' with his request to withdraw money. The state then terminates as a consequence of a final (dissolution) event such as 'the debiting of $300 from John Smith's account'.

Therefore, in the object paradigm a business process corresponds to a temporal part of an organization (decomposed into a set of events and states which are temporal parts of the process itself). A business process can in turn have substates; for example, particular activities of a process and so on. This correspondence between process/activity and state is not completely new to traditional behavioral modeling techniques and has a precedence in UML 1.x in which activity diagrams were considered a variation of state diagrams. Figure 3 illustrates the ontology of a business process as a 4D object part of an organization.

It must be remembered that ontologies acknowledge the existence of real world objects and systems. The ontology of a process that has already taken place consists of identifying the events and states of the process as illustrated above. However when representing business process models the modeler's aim is to provide a representation of all possible scenarios. These scenarios correspond to possible future states in which the organization can be. A business process model must cater for all foreseeable types of states (or possible worlds).

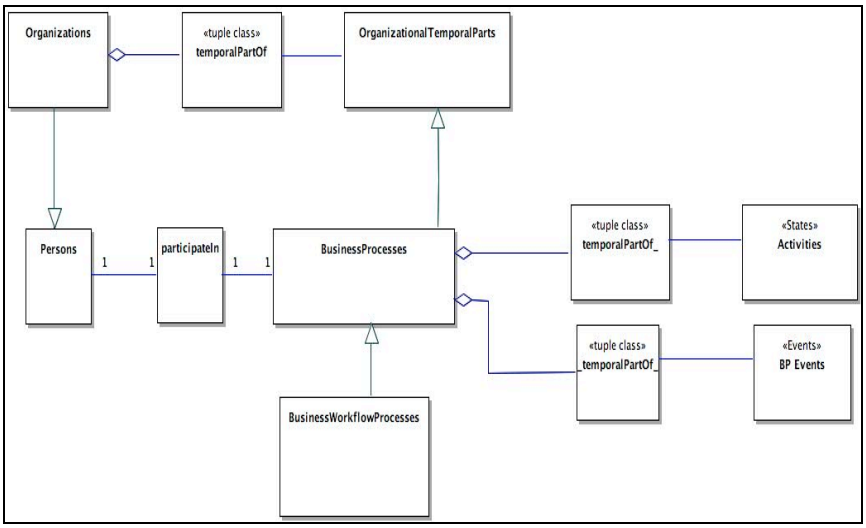

Fig. 3. Business Process Ontology (partial view)

3.2 Semantic Discovery Lifecycle

The Semantic Discovery Lifecycle (SDL) initiates with the procurement and organization of legacy sources and finishes with the production of business process patterns, which then become part of the pattern repository. The repository feeds into the Semantic Reuse Lifecycle. The phases of the SDL are as follows:

- *Procurement and Organization of Legacy Assets (POLA)*: SDL is a process of discovery; therefore it is necessary to derive the business process patterns from legacy assets that demonstrate the existence of certain types of models as well as

their generalized recurrence across multiple organizations. SDL, in this sense, is similar to the way scientific theories are discovered from scientific data. Only model types which have been previously and demonstrably adopted by organizations and/or workflow systems can be modeled and become part of the patterns repository. Therefore, acquiring legacy assets and organizing them in a repository is an essential initial step.

- *Segmentation of Legacy Assets (SLA)*: Before any type of semantic analysis of the legacy assets can take place, the assets need to be 'chunked' into workable fragments. For example, all documentation and models related to financial transactions of retail bank accounts can be collected together and fed into the next phase.
- *Semantic Analysis of BP Models (SA)*: This phase along with the following represent the core of SDL. In SA business process models are extracted from the legacy asset fragments. These models are typical process flow diagrams such as UML activity diagrams or BPMN diagrams. The elements of the process diagrams are then interpreted from an object paradigm perspective in order to derive more precise ontological models of the processes themselves.
- *Semantic Enhancement of BP Models (SE)*: This phase takes the ontological models created in SA and aims at generalizing them to existing patterns or to newly developed patterns.
- *Pattern Documentation (PD)*: The pattern(s) derived from a cycle of SDL are finally documented and catalogued in the patterns repository.

3.3 Semantic Reuse Lifecycle

The Semantic Reuse Lifecycle (SRL) is aimed at producing business process models with the support of the patterns discovered during the SDL. The phases of the lifecycle as illustrated in Figure 1 are purely indicative. An organization can adopt any business modeling process it prefers but such a process should then be tailored in order to include essential reuse activities such as matching the business requirements specifications with existing business process patterns and adapting such patterns (e.g., through specialization) to the specific requirement. The SRL is dependent on the SDL only in terms of the patterns that are produced by the SDL. The two lifecycles are, for all other purposes, autonomous and can be performed by different organizations. In this case the organization performing the SDL would be specialized in the management and supply of process patterns, while its clients would consume the discovered patterns. The typical phases of the SRL are as follows:

- *Requirements Analysis*: A given business problem is studied producing a set of business requirements specifications.
- *Matching of Patterns to Requirements*: Given the requirements produced in the previous phase, the requirements specifications are matched against existing business process patterns in order to identify patterns that can help to model and provide proven solutions to the requirements.
- *Pattern Specialization*: The patterns selected as possible template solutions to the specified requirements are then adapted to meet specific aspects of the problem space represented by the given requirements.

- *Model Production*: Models are produced as a solution to the business requirements.
- *Model Validation*: The models are validated (tested) against the business requirements until the solution provided is considered to be sufficiently adequate. At this stage it may be necessary to revisit the initial requirements if any omissions or amendments are identified. In this case the cycle is repeated.

4 Worked Example

The worked example presented in this section derives from the application of the SDL to legacy assets of three financial legacy systems. The three applications were produced by a large provider of business solutions and adopted by various important financial institutions worldwide. The specific financial domains that the systems are related to are, respectively, the retail banking, insurance and mortgage loans domains. The main aim of the worked example is to demonstrate how the semantic discovery process (SDL) is performed. The example is related to the creation of new product types in the financial industry. The preparatory phases of the SDL (POLA and SLA) will only be succinctly discussed here so as to provide the reader with sufficient information to understand the origin and nature of the legacy assets used and to dedicate most part of the worked example to the more significant phases of the SDL (i.e., SA and SE).

Preparatory Phases
The legacy assets used to derive the pattern of this example were (1) design and user manuals of the legacy systems and (2) knowledge acquired by interviews with two experts of the financial domain as well as of the legacy systems. The fragment analyzed here relates to the 'creation of new product types'. The fragment consists of all parts (chapters and/or sections) of the manuals providing information related to the generation or amendment of product types. The interview notes are also part of the fragment of legacy assets analyzed.

Semantic Analysis of BP Models
As mentioned previously, for SA to be conducted it is necessary to derive traditional process models from the fragment. Three process models were represented in BPMN. The three models (Figure 4) represent the workflow behavior that the legacy applications automate within the business processes of the respective organizations that have adopted this technology. Although similarities can already be noticed at this stage of the SDL, the next step consists of ontologically analyzing each element of the process models (e.g., events, tasks, etc.). This analysis enables the analyst to 'unbundle' or 'interpret' the semantic content of the process elements. This step is essentially a transformation from a traditional process paradigm to the object paradigm. The reason for undertaking such a transformation lies in the greater expressivity of the object paradigm. Expressing a model in a semantically richer and more precise paradigm facilitates the next phase of SE. Given limitations of space an extract of the semantic analysis is provided in Table 1.

Table 1 should be read as follows: for each modeling element (e.g., activity, event, etc.) of the legacy process models, that element commits to (or recognizes the existence of) the objects in the column labeled 'commits to'. This is fundamentally a tabular representation of an ontic commitment model (OCM) adapted from [20]. Similar tables are produced for the other two processes. The OCM shown in Table 1, although incomplete, highlights some interesting problems which also arise when semantically analyzing the other two process models:

1. What is the difference between a product template and a product type? Are they the same class? Do instances of the former represent states of instances of the latter?
2. The process system recognizes 'categories' such as demand, savings, loan, etc. These are types of account products that are 'universally' recognized. What is the difference between these product types and those created and marketed by banks?
3. When a product template is changed creating a new 'version', does this version correspond to a new product type or a state of an existing product type?

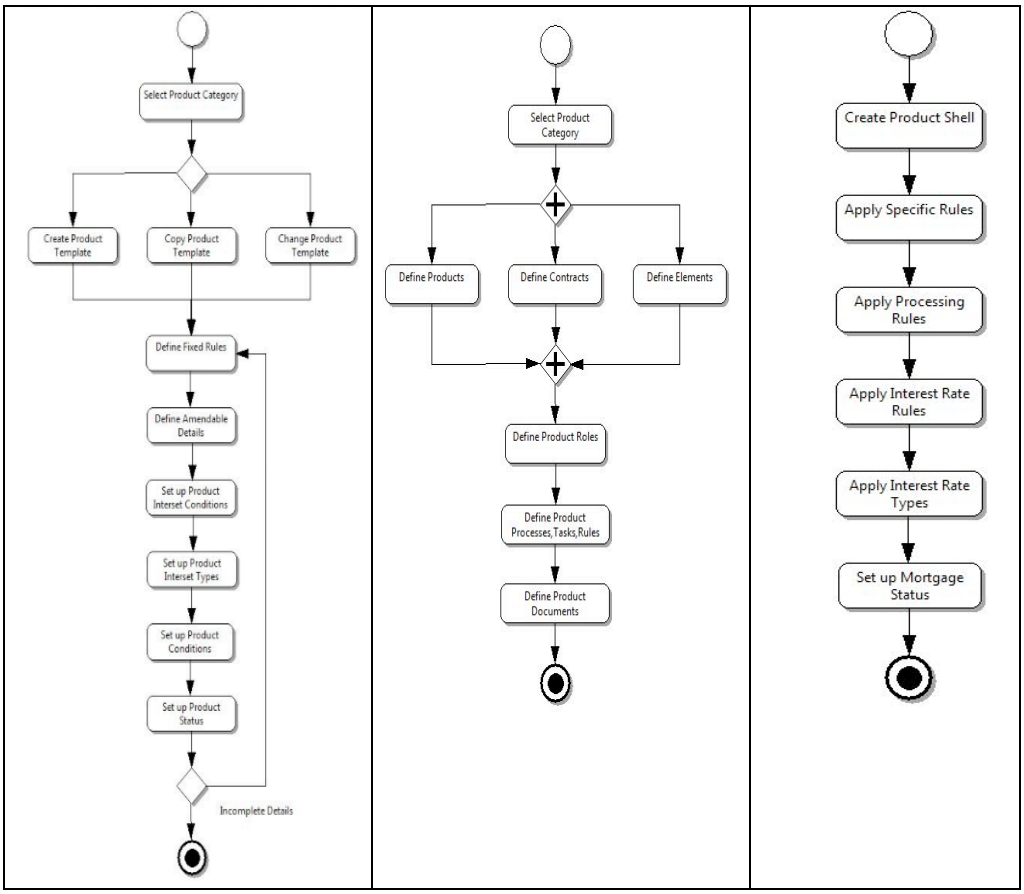

Fig. 4. Legacy Process Models

Table 1. Tabular Ontic Commitment Model (adapted from [20]).

	Activity/Event	Commits To	Type of Object
1	Start event	Decision to Create a New Product Type	Event
2	Select a product category	Template Creation	Creation Event
3	Create product template		
4	Copy product template		
5	Change product template	Product Type Version	State
6	Define fixed rules	PTemplate Fixed Rules Def Stage	State
7	Define amendable rules	PTemplate Optional Rules Def Stage	State
8	Set up product interest conditions	PTemplate Interest Conditions Def Stage	State
9	Set up product interest types	PTemplate Prod Interest Type Def Stage	State
		Interest Types	Class
10	Set up product conditions	PTemplate Product Conditions Def Stage	State
11	Set up product status	Product Type Creation	State
		Draft State	State
		Live State	State
		Suspended State	State

Semantic Enhancement of BP Models

The SE phase is aimed at identifying limitations of and improvements to the models produced in SA. In particular SE identifies generalization-specialization relationships between processes. The steps that are carried in SE can be summarized as follows:

- The OCM are compared in order to identify: (1) similarities between the objects that the legacy models commit to, (2) generalize similar types of process states and (3) domain objects that can improve the semantic quality of existing ontologies. The example questions that were raised in the previous subsection are instrumental toward achieving these intended goals.
- Produce a generalized version of the original process models.

From the semantic analysis of the three process models the following similarities are identified:

1. In all cases the product type is termed as product template or shell raising confusion as to the difference is between the two. It would seem that the legacy systems do not make an explicit distinction between the template (which is a 'mould' of possible types) and product type itself. A template or shell can be considered as a model or prototype of a developing product type, which, however, does not exist until the 'going live' date is reached. With this event happening then a product type begins.

The 'product type creation' workflow is therefore composed of two significant temporal parts: (1) a state in which the product template is being defined and (2) an event ('going live') which gives birth to the product type. From that moment onward specific products instances of the newly created product type can be offered to the organization's customers.

The 'product template definition' represents a state in which the organization (including the computer systems that automate the workflow) is. Conceptually a product template is generated and persisted as well once the workflow terminates. The template can be used again as a 'mould' for future product types.

2. Categories of products, such as ‘demand’, ‘savings’, etc. have been termed as ‘analytic product types’ as opposed to ‘intentional product types’. The main difference between the two classes is that the former represents classes of products as defined by the financial industry at large. The latter, instead, represents classes of products that companies create to be marketed to consumers/clients: for example, ‘Mega Bank’s Gold Current Account’.
3. From the previous SA phase it was noted that the model representation allow for new versions of existing product types to be created. The semantic interpretation of the legacy data has led us to conclude that: (a) a version is not a new product type but a state of an existing one; (b) individual products of the previous version cannot be released to customers once the once the new version enters its ‘live’ substate.

The model related to point 1 above is represented in Figure 5. From the generalization process of the activities/states derived from the three process diagrams the following have been derived as generalized state classes: ‘Assign Analytic Product Type’, ‘Definition of Mandatory Characteristics’, ‘Definition of Optional Characteristics’ and ‘Set Status’.

Patterns Documentation

The pattern(s) derived in SE are then documented in a standardized format. Both the document and SE models are archived for future use in the patterns repository.

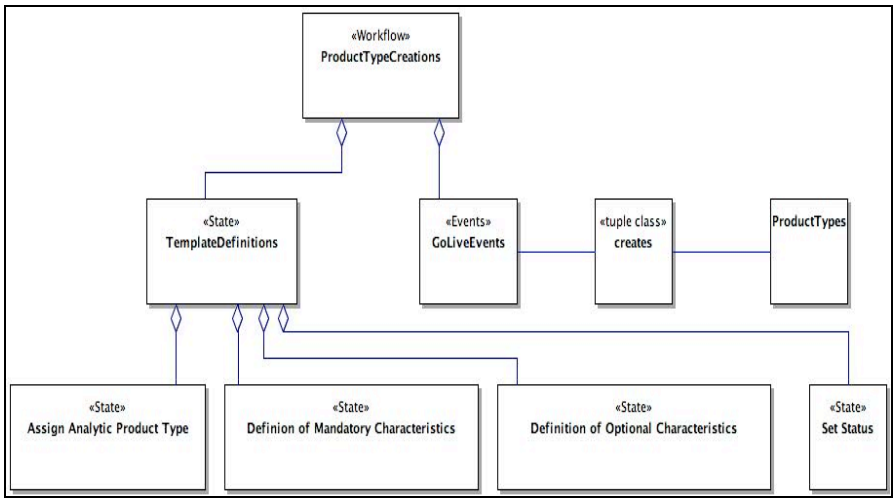

Fig. 5. Generalized Ontological Pattern for the Creation of New Product Types

5 Conclusion and Future work

This paper presented a framework for the semantic discovery and reuse (SDR) of business process patterns. The framework defines a dual lifecycle model. The first lifecycle is aimed at deriving business process patterns from legacy content through the use of ontologies. The second lifecycle is aimed at business modeling and reuses the patterns defined in the previous lifecycle.

The approach to semantic discovery was demonstrated in its salient features through a worked example. The example was based on processes derived from three legacy systems. The SDR framework overcomes two limitations of previous research on business process patterns. Firstly, the workflow patterns defined by [11] model common control structures of workflow languages are not aimed at modeling generic processes of a business domain (like an industrial sector). Secondly, the patterns research community to date has dedicated limited attention to the process of patterns discovery. The unique features of the SDR framework are its dual lifecycle model, its use of semantics (modeled with ontologies based on a 4D perspective) and the grounding in real world legacy models and data to derive the patterns. This last point is of particular importance because it underlines the fact that the modeled patterns must be based on evidence of their actual existence.

The work presented here is ongoing. The following phases of our research will be to: (1) continue discovering business process patterns from legacy systems; (2) continually test the existing patterns against legacy models and data and (3) define a maturity model of business process patterns based on the type of testing that the patterns have undergone (e.g., tested against one legacy system, against multiple systems of one domain and, finally, multiple systems across multiple domains).

References

1. Hammer, M., & Champy, J.: Reengineering the Corporation: A Manifesto for Business Revolution. Rev. edn. Brealey, London (2001)
2. Lindsay, A., Downs, D., Lunn, K.: Business Processes—Attempts to Find a Definition. Information and Software Technology, 45**,** 1015-1019 (2003)
3. Eriksson, H., & Penker, M.: Business Modeling with UML: Business patterns at work. John Wiley & Sons, New York, Chichester (2000)
4. Kaisler, S. H.: Software Paradigms. John Wiley & Sons, Hoboken, N.J. USA (2005)
5. Alexander, C., Ishikawa, S., Silverstein, M. et al.: A Pattern Language: Towns, Buildings, Construction. Oxford University Press, New York (1977)
6. Beck, K., & Cunningham, W.: Using Pattern Languages for Object-Oriented Program. Workshop on Specification and Design for Object-Oriented Programming
7. Coad, P.: Object-Oriented Patterns. Commun ACM, 35, 152-159 (1992).
8. Coad, P., de Luca, J., Lefebvre, E.: Java Modeling Color with UML: Enterprise Components and Process. Prentice Hall PTR, Upper Saddle River, NJ, USA (1999)
9. Gamma, E., Helm, R., Johnson, R. et al.: Design Patterns: Elements of Reusable Object-Oriented Software. Addison-Wesley, Reading, Mass. (1995)

10. Hay, D. C.: Data Model Patterns: Conventions of Thought. Dorset House Pub. New York (1996)
11. Fowler, M.: Analysis Patterns: Reusable Object Models. Addison-Wesley Object Technology Series. Addison Wesley, Menlo Park, Calif. (1997)
12. van der Aalst, W.M.P., ter Hofestede, A.H.M., Hofestede, Kiepuszewski, B., and Barros, A.P.: Workflow Patterns. Distrib. Parallel Databases, ACM, 14, 5-51 (2003)
13. Thom, L. H., Iochpe, C., Reichert, M.: Workflow Patterns for Business Process Modeling. Proceedings of Workshops and Doctoral Consortium of the 19th International Conference on Advanced Information Systems Engineering (CAiSE), pp. 349-358, Tapir Academic Press, Trondheim, Norway (2007)
14. van der Aalst, W.M.P., ter Hofestede, A.H.M., Hofestede, Kiepuszewski, B.: Advanced Workflow Patterns. 7th International Conference on Cooperative Information Systems (CoopIS 2000), pp. 18, Springer-Verlag, Berlin (2000)
15. Di Dio, G.: ARWOPS: A Framework for Searching Workflow Patterns Candidate to be Reused. Internet and Web Applications and Services. ICIW '07. Second International Conference, pp. 33, IEEE CNF, Mauritius (2007)
16. Prieto-Daz, R.: Domain Analysis: An Introduction. SIGSOFT Software Engineering Notes, 15, 47-54 (1990)
17. Foreman, J.: Product Line Based Software Development- Significant Results, Future Challenges. Software Technology Conference, Salt Lake City, UT (1996)
18. Partridge, C.: Business Objects Re-Engineering for Re-use. 1st edn. Butterworth-Heinemann College (1996)
19. Sider, T.: Four-Dimensionalism: An Ontology of Persistence and Time. Oxford (2003)
20. Daga, A., de Cesare, S., Lycett, M. et al.: An Ontological Approach for Recovering Legacy Business Content. In Proceedings of the 38th Annual Hawaii International Conference on System Sciences (HICSS'05), pp. 224a, IEEE Computer Society Press, Los Alamitos, California (2005)