The RiverFish Approach to Business Process Modeling: Linking Business Steps to Control-Flow Patterns

Devanir Zuliane[1], Marcio K. Oikawa[1], Simon Malkowski[2], José Perez Alcazar[1], and João Eduardo Ferreira[1]

[1] Department of Computer Science, University of São Paulo,
Rua do Matão 1010, 05508-090 São Paulo, Brazil
{devanirz,koikawa,jperez,jef}@ime.usp.br
[2] Center of Experimental Research in Computer Systems,
Georgia Institute of Technology,
266 Ferst Drive, 30332-0765 Atlanta, USA
simon.malkowski@cc.gatech.edu

Abstract. Despite the recent advances in the area of Business Process Management (BPM), today's business processes have largely been implemented without clearly defined conceptual modeling. This results in growing difficulties for identification, maintenance, and reuse of rules, processes, and control-flow patterns. To mitigate these problems in future implementations, we propose a new approach to business process modeling using conceptual schemas, which represent hierarchies of concepts for rules and processes shared among collaborating information systems. This methodology bridges the gap between conceptual model description and identification of actual control-flow patterns for workflow implementation. We identify modeling guidelines that are characterized by clear phase separation, step-by-step execution, and process building through diagrams and tables. The separation of business process modeling in seven mutually exclusive phases clearly delimits information technology from business expertise. The sequential execution of these phases leads to the step-by-step creation of complex control-flow graphs. The process model is refined through intuitive table and diagram generation in each phase. Not only does the rigorous application of our modeling framework minimize the impact of rule and process changes, but it also facilitates the identification and maintenance of control-flow patterns in BPM-based information system architectures.

Keywords: business process management, conceptual schema, control-flow patterns, information systems, process modeling.

1 Introduction

With the advent of internet services a new way of making business has been introduced in many important markets. Today, numerous transactions that could

E. Bertino and J.B.D. Joshi (Eds.): CollaborateCom 2008, LNICST 10, pp. 179–193, 2009.

previously only be carried out in person (e.g., purchasing books, paying electricity bills, or participating in auctions) are readily available online. In order to remain competitive in the market, modern enterprises must adapt to this new way of making business in an effective way. However, the gathering of conceptual knowledge involved in the business processes of an enterprise is highly dependent on a clearly formulated acquisition framework and its rigorous enactment [7]. In other words, implementing business processes as workflows in information systems without thorough understanding of Business Process Management (BPM) methodology may result in severe problems.

This new business environment imposes particularly great challenges on large corporations with highly complex information system. Characteristically, these systems are composed of heterogeneous components that require collaboration to achieve high-level objectives. In this context, one of the major problems of classical business process languages is their implicit representation of hierarchical business step classification. Commonly, business steps are defined as atomic units of business processes, and they are inherently difficult to identify and classify correctly. We address this challenge through conceptual modeling, which is well established for mapping models and their relationships. Conceptual schema link the understanding of an organizational structure on various levels of abstraction and can incorporate different modeling concepts (e.g., the entity-relationship model or the UML class model [5]). Hence, they allow the specification of concept hierarchies and offer the necessary expressiveness for hierarchical business rule definition. Consequently, our approach introduces an explicit domain-layer that is well-structure, independent of framework, and independent of actual workflow implementation.

The main contribution of this paper is a new approach to business process modeling with direct integration of conceptual design. As an alternative to classical business process modeling, we base our work on a conceptual schema that enables the hierarchical classification of business process building blocks. This structured derivation of business steps is used for the identification of control-flow patterns in the underlying business processes. Concretely, we formulate the *RiverFish Conceptual Schema* based on our previous work on the RiverFish architecture [7, 10]. We present a methodical step-by-step identification of underlying business steps, control-flow patterns, and business processes. The output of this approach constitutes the key input of successful business process implementation.

This five section paper is organized as follows. Section 2 discusses related works in the areas of business process and workflow management. In Section 3 we introduce our methodical foundation and describe the proposed conceptual schema along with the notion of control-flow patterns. Section 4 presents the guidelines that lead to the identification of control-flow patterns and business processes starting from the conceptual schema. We exemplify the application of this method in a simplified real collaborative information system case study. Finally, Section 5 concludes our findings and touches upon ongoing research.

2 Related Work

The representation and execution of business rules is a widely researched area. The predominant approaches are the design of specialized workflow languages [2, 3, 9] and large unifying methods of effective management of business processes [1, 13]. A workflow management system defines, creates, and manages the execution of workflows through the use of software running on one or more platforms [15]. Such systems are capable of interpreting the definition of processes, interacting with the participating users, and making (potentially necessary) external calls to tools and applications. Weske presents a comprehensive overview of the main concepts of BPM [13] . The emphasis lays on business process modeling, orchestration, and choreographies as well as business process properties, such as data dependencies and structural soundness. However, the derivation phase of business processes remains largely ambiguous in these references. They do not offer structured details about the identification of control-flow patterns or the composition of business steps into business processes. The methods used to quantify the actual business process flow in an enterprise are not explicitly discussed in any of these works. To this date, the identification of control-flow patterns has largely relied on the know-how of highly specialized and highly paid domain experts.

The necessity for rigorous methodologies in the BPM domain is well established and well understood. Hofstede et al. established three reasons for using the Petri net formalism in workflow specification [9]. Petri nets are formal; associate sophisticated analysis techniques with workflows; and are based on states rather than events. A detailed description of control-flow patterns based on Petri nets can be found in [3]. Nevertheless, the specification of all dependencies in control-flows remains a complex task and has led to the development of specifications such as Yet Another Workflow Language (YAWL) [4]. The latter addresses these challenges explicitly and emphasizes formal semantics in the transition systems.

Van der Aalst et al. have previously identified four different basic mechanisms in process management: sequence, selection, parallelism and iteration [3]. All of them are common in practice, and the authors infer that comprehensive real-world workflow functionality can be exhaustively modeled using these four mechanisms. The authors' main goal is to enable a comparative analysis between all major languages for business process specification and aid in business process modeling. Accordingly, they classify the control-flow patterns into six categories: basic control-flow; advanced branching and synchronization; structural; multiple instance; state-based; and cancellation. This work led to the foundation of the Workflow Patterns Initiative (`www.workflowpatterns.com`) at the 4th International Conference on Cooperative Information Systems (IFCIS 99). Today, many academic and technological results have contributed to the growth of the control-flow pattern repository, which spans over 100 patterns in control-flow, data and resource approaches. These patterns helped to define evaluations of UML 2.0 Activity Diagrams and Business Process Modeling Notation (BPMN) [14]. Although these efforts are closely related to the research presented in this paper, our work is orthogonal. While we focus on the methodological identification of

control-flow in a real information system environment, the previously discussed approaches deal with the comparison and definition of workflow structures. Although control-flow patterns have had a significant impact on the field of workflow technology, their derivation and design remain a loosely structured process in the cited references. In general, there has been very little concern with the derivation and classification of business processes and rules. Even large initiatives such as BPMN [14] do not address this issue.

3 Foundation

3.1 Control-Flow Patterns

The notion of categorizing workflow management systems through patterns identified by four basic mechanisms in the process structure was introduced by van der Aalst et al. [3]. The authors argue that their systematic methodology enables reasoning about suitability and expressiveness of workflow frameworks. We utilize this method and adopt van der Aalst's approach in our work. Therefore, understanding the implications of control-flow patterns is an important prerequisite to understanding the RiverFish Conceptual Schema. Unfortunately, it is not an easy task to provide a thorough explanation using only few pages. Interested readers should refer to the original reference [3] for a comprehensive presentation and a detailed discussion. We confine ourselves to a sampled overview in Figure 1 and three brief examples in the following.

1. Patterns 4 and 16 in Figure 1 are called *Exclusive* and *Deferred Choice*, respectively. If the choice is exclusive, one of several activity branches is chosen based on decision or on workflow data. If the choice is deferred, it is not made explicitly. All alternatives are offered to the environment, which chooses only one of them for execution.
2. Pattern 8 in Figure 1 represents the *Multi-Merge*. This pattern constitutes "[...] a point in a workflow process where two or more branches re-converge without synchronization. If more than one branch gets activated, possibly concurrently, the activity following the merge is started for every activation of every incoming branch" [3].
3. Pattern 9 in Figure 1 is called the *Discriminator*. According to [3] it represents "[...] a point in a workflow process that waits for one of the incoming branches to complete before activating the subsequent activity. From that moment on it waits for all remaining branches to be completed and ignores them".

3.2 RiverFish Conceptual Schema

The RiverFish Conceptual Schema is founded on the concepts of the RiverFish architecture [7, 10] and on the classification of business rules proposed by von Halle [8]. A business rule is a declaration that restricts certain aspects of a business in order to define its structure or control its behavior [12]. Business rules are inherent to any type of business process and can be classified in the following four types [8].

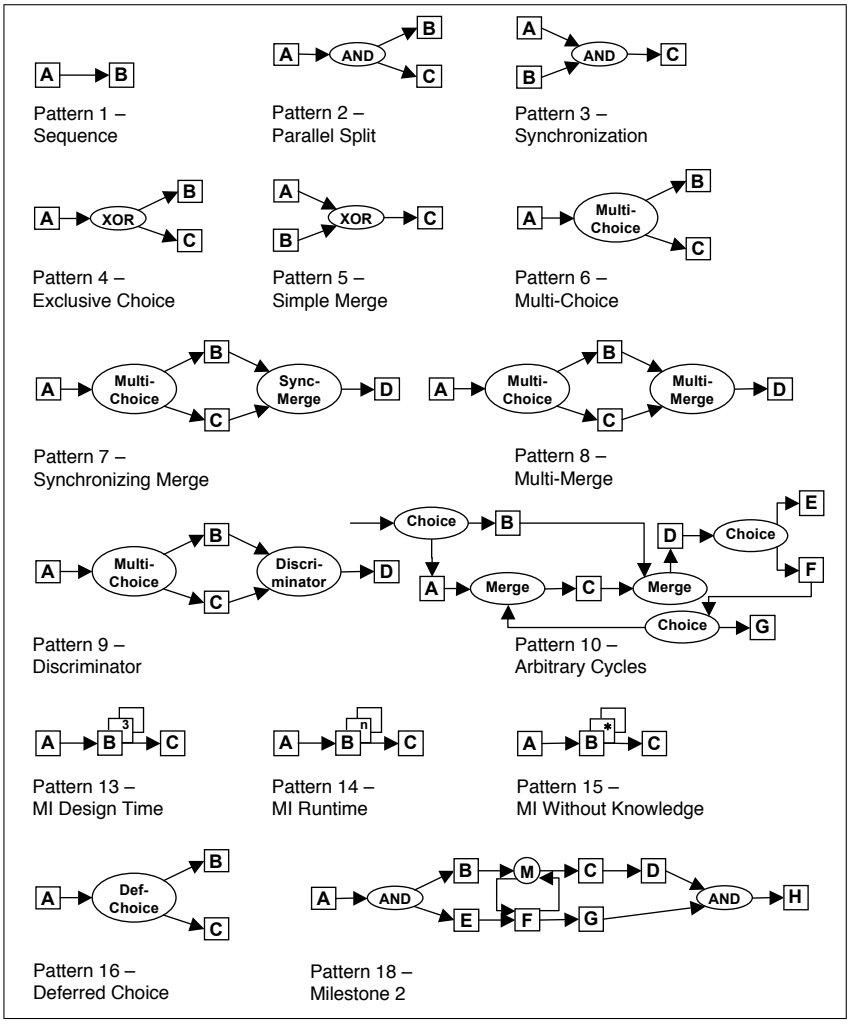

Fig. 1. Sample graphical representations of fifteen control-flow patterns adapted from [3]

1. A *constraint* is a declaration that expresses an unconditional circumstance, which must be either true or false (e.g., to be selected, a company must be authorized).
2. An *action enabler* is a declaration that verifies conditions, and some action is initiated in case the conditions are true (e.g., if the contributor's record is not on file, then execute the filling of the record).
3. A *computation* is a declaration that supplies an algorithm to calculate a term after verifying some conditions (e.g., if there is a record for the selected model, then consider the initial numbering to be the increment of the previous number).

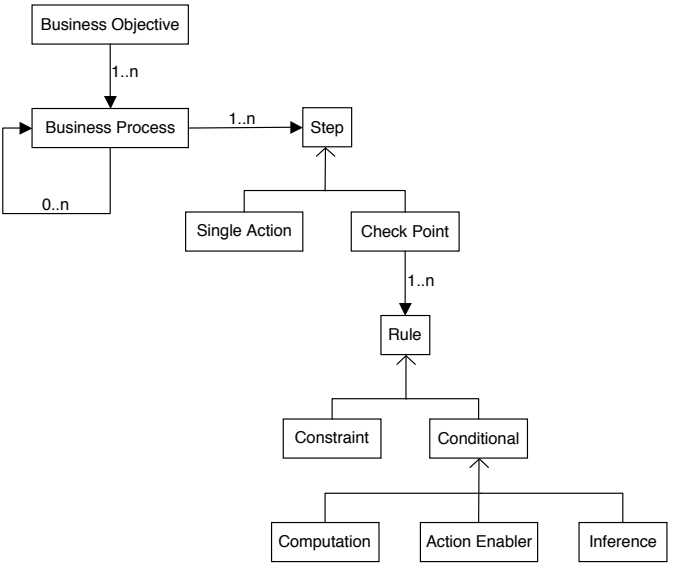

Fig. 2. The RiverFish Conceptual Schema, which extends the original RiverFish approach

4. An *inference* is a declaration that tests a condition and declares a new fact if it finds the condition to be true (e.g., if a client is a regular client, then he gets a discount on his purchases).

The RiverFish architecture represents request processing through modeling and execution of business steps under a common business goal [10]. However, RiverFish itself does not detail the differences between rules, validations and constraints. As such, the categorization of business rules in a given process remains ambiguous. Therefore, the RiverFish Conceptual Schema extends the definition of the business process *check point* using a common classification of business rules [8]. Careful evaluation of this classification approach yields the understanding of the inherent structural similarity between *action enablers*, *computations*, and *inferences*. Their characteristic similarity can be expressed in an intuitively clear format; i.e., `If <condition>, Then <action>`. Consequently, it is possible to group all types of business rules into solely two categories from the conceptual point of view; i.e., *constraints* and *conditionals*. Constraints can be modeled and defined by a single validation; i.e., "a certain condition must be true". Hence, this validation can easily be represented using first order logic or logic based on frames. On the other hand, the only difference between rules that are conditionals (i.e., action enabler, computation and inference) lays in the action to be executed in case the condition is true. The resulting schema is illustrated in Figure 2.

By comparing the conceptual schema approach to the initial RiverFish architecture, it becomes apparent that the adaptation occurred only at the rule level, preserving the entire previous structure. This is justified by the fact that the

Table 1. The characteristics and details of all business rule types

Rule Type	Conditional Type	Structure	Detail
Constraint	%	`<condition> = true or false`	`<condition>` is a logical expression
Conditional	Action Enabler	`If <condition> Then <action>`	`<action>` executes a procedure
	Computation	`If <condition> Then <action>`	`<action>` calls an algorithm for computation
	Inference	`If <condition> Then <action>`	`<action>` calls an execution to enable another rule

original architecture proposed a generic approach to controlling the processing of business rules. The details of all rule types of are given in Table 1.

4 From Business Steps to Control-Flow Patterns

To illustrate the proposed guidelines we present a simplified case study with an appropriate level of complexity to allow some scenario diversity. The electronic AIDF (a Portuguese acronym for Authorization to Print Fiscal Documents) is a large authorization system. Contributors make requests to the Finance Ministry of the state of São Paulo (SEFAZ) to print authorized fiscal documents. SEFAZ receives all requests from all contributors and chooses to approve or to decline them after a selective analysis. Because of space constraints, details about this process were omitted in this paper. Tables 2 to 5 present the main aspects of the AIDF case study.

Despite the RiverFish Conceptual Schema providing the hierarchy of business steps, checkpoints, and rules, this is not sufficient to identify the execution ordering for the entire business process. More concretely, there is a transition gap between conceptual schema and control-flow patterns. We provide this missing link through a phase model summarized in Figure 3. The main characteristics of the guidelines are listed in the following.

1. Separation of business process modeling in seven phases that characterize and delimit the business and information technology expertise;
2. Step-by-step application of these phases to generate the complex control-flow graphs;
3. Derivation of the process model through tables and diagrams generated in each phase.

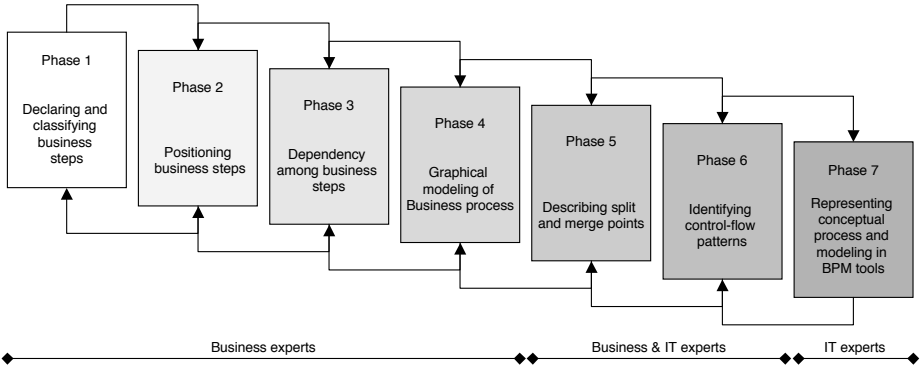

Fig. 3. Guidelines for business process conceptual modeling

In the rest of this section we give details for each of the seven phases and describe them in the context of the AIDF case study.

4.1 Declaring and Classifying Business Steps (Phase 1)

The first phase has the objective to declare and to classify the business steps of the RiverFish Conceptual Schema described in Section 3.2. The declaration of business steps seeks to turn all the steps that compose a business process explicit without worrying about the identification of control-flow structures (such as operators of sequences and alternative compositions). Table 2 presents an example of a purchasing business process with several steps composed in an arbitrary order.

After the declaration of business steps, they are classified into single actions and checkpoints. Checkpoints can be either constraints or conditional rules. In the business process example presented in Table 2, step D (i.e., "Validate driver ID") is a constraint rule because the validation is impossible without ID following

Table 2. Declaration of business steps

Step	Declaration
A	Generate data form
B	Fill out and send data form
C	Store data form
D	Validate driver ID
E	Update user data
F	Generate list of products for sale
G	Prohibit sale for younger than 18
H	Inform customer's fidelity
I	Calculate saving for customer
J	Calculate the purchase total
K	Finish the purchase

Table 3. Classification of business steps

Step	Declaration	Classification
A	Generate data form	Single action
B	Fill out and send data form	Single action
C	Store data form	Single action
D	Validate driver ID	*Constraint*
E	Update user data	Single action
F	Generate list of products for sale	Single action
G	Prohibit sale for younger than 18	*Constraint*
H	Inform customer's fidelity	Single action
I	Calculate saving for customer	Single action
J	Calculate the purchase total	Single action
K	Finish the purchase	Single action

Table 4. Step recomposition

Step	Declaration	Classification
A	Generate data form	Single action
B	Fill out and send data form	Single action
C	Store data form	Single action
D	Validate driver ID	*Constraint*
E	Prohibit sale for younger than 18	*Constraint*
F	Re-submission in case of invalid ID or younger than 18	Single action
G	Update user data	Single action
H	Generate list of products for sale	Single action
I	Inform customer's fidelity	Single action
J	Calculate saving for customer	Single action
K	Calculate the purchase total	Single action
L	Finish the purchase	Single action

a purchase. On the other hand, step H (i.e., "Inform customers fidelity") is a conditional rule. If step H is true then the computation step I (i.e., "Calculate saving for customer") is taken. Step G is another constraint-checkpoint because it prohibits the sale based on age limit. The classification of business steps directly aids in the business process control-flow ordering. Nevertheless, during the first phase only the constraint steps are clearly identified. In our example, all other steps are classified as single actions in order to facilitate the notation. Table 3 presents the resulting classification of business steps belonging to Table 2.

4.2 Positioning Business Steps (Phase 2)

A step classified as constraint check point can generate interrupts in the execution of business control-flows. This kind of check point only allows true responses

to continue. The main goal in this phase is repositioning constraint-checkpoints whenever possible to the beginning of a control-flow. In our case study, the new step F is added for the handling of a false answers. Its sole purpose is to jump back to the start. This step repositioning reduces the effects caused by false answers at constraint check points. Table 4 shows the effect of step repositioning in the case study.

4.3 Dependency among the Business Steps (Phase 3)

This phase identifies the interdependencies among all steps. Table 5 presents the business step dependencies in our example. The notation X→Y defines Y as dependent on X. According to Table 5, steps C, D, E, and G have more than one dependency. Hence, they are called split points.

4.4 Graphical Modeling of Business Process (Phase 4)

In this phase, we use four basic connectors (illustrated in Figure 4) to generate the graphical modeling of business processes. With the help of these operators each business step has to be linked according to its dependency relationships. Figure 5 shows the results of this phase in the case study. This visual insight clearly reveals sequential and parallel business steps. However, it does not show the details for selection and synchronism mechanisms determining split and merge points. Capturing and declaring the behavior of split and merge points is fundamental to identifying and classifying underlying control-flow patterns. Nevertheless, in this phase we simply confine the labeling to integer numbers and leave the detailed identification for the next phase.

Table 5. Examples of business step dependencies

Step	Declaration	Classification	Dependency
Start			(Start)→A
A	Generate data form	Single action	A→B
B	Fill out and send data form	Single action	B→C
C	Store data form	Single action	C→D and C→E
D	Validate driver ID	*Constraint*	D→G or D→F
E	Prohibit sale for younger than 18	*Constraint*	E→G or E→F
F	Re-submission in case of invalid ID or younger than 18	Single action	F→(Start)
G	Update user data	Single action	G→H or G→J
H	Generate list of products for sale	Single action	H→I
I	Inform customer's fidelity	Single action	I→L
J	Calculate saving for customer	Single action	J→K
K	Calculate the purchase total	Single action	K→L
L	Finish the purchase	Single action	L→(End)
End			

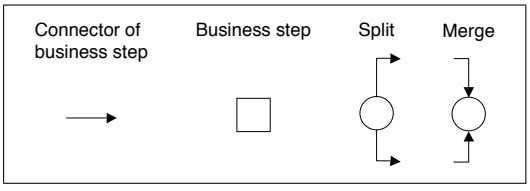

Fig. 4. Four basic connectors

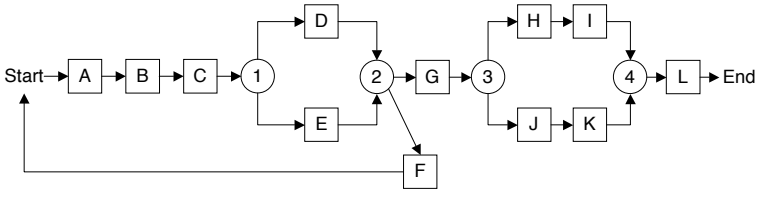

Fig. 5. Control-flow with dependencies

4.5 Describing the Split and Merge Points (Phase 5)

In general, split and merge points are classifieded by their selection mechanism. For instance, there are many possibilities for split execution of steps D and E.

1. D and E initialize in any order;
2. D and E initialize simultaneously;
3. E initializes after D;
4. D initializes after E.

The same applies to merge synchronization after the execution of D and E. There are the following four possibilities.

1. Waiting for the simultaneous execution of D and E to enable F or G;
2. Waiting for the execution of D or E enables F or G;
3. Waiting for the execution of D enables F or G without waiting for E;
4. Waiting for the execution of E enables F or G without waiting for D.

Table 6 describes the actual behavior for each point in our case study.

4.6 Identifying Control-Flow Patterns (Phase 6)

For Phase 6, it is important to understand the behavior of the control-flow patterns introduced in Section 3.1 and depicted in Figure 1. Using these patterns and the descriptions in Table 6, each split and merge point is matched to the best-fitting control-flow pattern. The resulting matching is shown in Table 7.

After identification, the corresponding control-flow pattern numbers are added to the dependency graph as shown in Figure 7. This graphical representation shows

Table 6. Behavior declaration for split and merge points

Label	Split	Merge	Behavior Declaration
1	√		Steps D and E can be executed simultaneously or in any order.
2		√	After executing D and E, it is possible to enable G. If either D or E are not true, F is enabled.
3	√		We can choose steps H after J, J after H, or the simultaneous execution of H and J and any possible combination of I and K afterwards.
4		√	If I, K, or I and K execution has completed, it is possible to enable L.

Table 7. Identification of control-flow patterns

Label	Control-flow Pattern	Number
1	Parallel split	Pattern 2
2	Synchronizing merge	Pattern 7
3	Multi choice	Pattern 6
4	Multi merge	Pattern 8

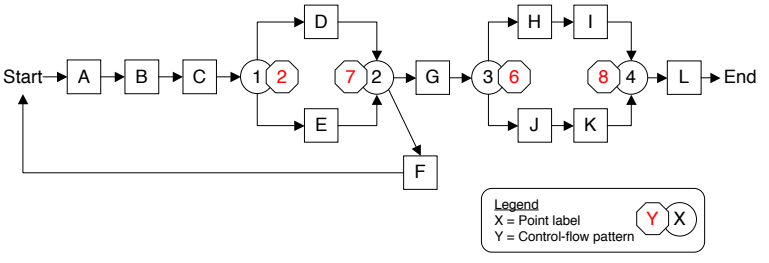

Fig. 6. The final result of the business processes modeling

the final output of the classification and identification process. These business processes can now be used as input of the implementation in information systems.

4.7 Representing Process Modeling in BPM Tools (Phase 7)

The implementation of the case study in phase 7 was carried out using BPM tools; however, it is out scope of this paper to show all details of this implementation. Instead, Table 8 lists all steps that were part of the actual implementation. All described steps are part of the authorization process for each AIDF request. The alias in the first column of Table 8 was established to facilitate the rule

Table 8. Steps of the AIDF business process

Alias	Description	Type
R1	The selected print shop must be `active` in SEFAZ;	Constraint
R2	The CNPJ of the print shop must be `valid`;	Constraint
R3	If the request is centralized, put the appropriate rules into action;	Action enabler
R4	The selected print shop must be `active` in SEFAZ;	Constraint
R5	The contributor must be `active` in SEFAZ;	Constraint
R6	If the contributor does not have a concluded record, verify that his record is on file;	Inference
R7	If the contributor has a record on file that is not concluded, execute the conclusion of the record;	Action enabler
R8	The selected print shop must be certified for the selected document;	Constraint
R9	If the models for the selected documents are allowed by the contributors National Classification of Economic Activities (CNAE), execute rule R10;	Inference
R10	If there is a record for the filtered models and the emission process is electronic, load the information on model, series, and subseries that are recorded in the Electronic System of Data Processing (SEPD);	Action enabler
R11	If there is a record for the filtered models, and the emission process is *not* electronic, load the information on the series and subseries;	Action enabler
R12	If there is a record for the selected model, series, and subseries, the initial Number equals the last printed number incremented by one;	Computation
R13	If the amount of copies informed is lower than the default amount recorded for the chosen model, execute rule R14;	Inference
R14	The contributor must have a justification to request unusual number of copies;	Constraint
R15	The operation field must be formatted with the default of the chosen model;	Constraint
R16	When emitted using the manual process, the fiscal documents in series B, C, D, and F must inform the destination of the subseries per operation as per the table;	Computation
R17	If the type of print-out is a PAD, the information Receipts per receipt pad should equal 20, 25, 30, 40, 50, or 100;	Computation
R18	The Amount of receipt pads should be equal to the amount of fiscal documents divided by the number of receipts per receipt pad;	Constraint
R19	If the counting process is `simple`, message one must be printed into the document;	Computation

Table 8. (*continued*)

Alias	Description	Type
R20	If the counting process is `electronic`, messages two and three must be printed onto the document;	Computation
R21	If the counting process is `electronic` and the request is `distributed`, message four must be printed onto the document;	Computation
R22	If the previous process executes `ok`, make the final number equal to the initial number plus the requested amount minus one.	Computation

manipulation. All rules were identified, grouped together, and classified according to von Halle [8]. The classification is listed in the third column of the table.

5 Conclusion

This paper has presented an alternative for classification of business processes and control-flow pattern identification, which constitutes the starting point for the implementation of business processes. We have adopted an approach based on conceptual modeling through conceptual schemas. This allows representing, analyzing, and documenting business processes in an application independent layer. Through the RiverFish Conceptual Schema, we avoid the coupling of rules and processes with code. This facilitates the adaptation and the handling of rules and processes in the actual application.

Another benefit of this approach is the formal representation of business rules that allows their systematic manipulation and automation. The methodology proposed in this work divides the complex task of business process modeling into seven phases and clearly delimits the actions taken by experts from different areas (i.e., business or IT). A step-by-step discovery of business dependencies allows identification of control-flow patterns, and resolves the complex identification of control-flows in business processes.

Our ongoing work includes the development of semi-automated algorithm planners and pattern recognition techniques for identifying control-flow patterns in business process modeling.

Acknowledgment

This work has been partially supported by FAPESP (São Paulo State Research Foundation) and CAPES (Brazilian Coordination for Improvement of Higher Level Personnel).

References

1. ABLE Project Team IBM T. J. Watson Research Center: ABLE Rule Language, User's Guide and Reference, Version 2.0.1 (October 2003)
2. van der Aalst, W.M.P., Berens, P.J.S.: Beyond workflow management: product-driven case handling. In: GROUP 2001: Proceedings of the 2001 International ACM SIGGROUP Conference on Supporting Group Work, pp. 42–51. ACM, New York (2001)
3. van der Aalst, W.M.P., ter Hofstede, A.H.M., Kiepuszewski, B., Barros, A.P.: Workflow patterns. Distributed and Parallel Databases 14(1), 5–51 (2003)
4. van der Aalst, W.M.P., ter Hofstede, A.H.M.: YAWL: Yet another workflow language. Inf. Syst. 30(4), 245–275 (2005)
5. Batini, C., Ceri, S., Navathe, S.B.: Conceptual database design: an Entity-relationship approach. Benjamin-Cummings Publishing Co., Inc., Redwood City (1992)
6. Braghetto, K.R., Ferreira, J.E., Pu, C.: Using control-flow patterns for specifying business processes in cooperative environments. In: Cho, Y., Wainwright, R.L., Haddad, H., Shin, S.Y., Koo, Y.W. (eds.) SAC, pp. 1234–1241. ACM, New York (2007)
7. Ferreira, J.E., Takai, O.K., Pu, C.: Integration of collaborative information system in internet applications using riverfish architecture. In: CollaborateCom. IEEE, Los Alamitos (2005)
8. Halle, B.V.: Business Rules Applied: Building Better Systems Using the Business Rules Approach. John Wiley & Sons, Inc, New York (2001); Foreword By-Ronald G. Ross.
9. Hofstede, A.H.M.T., Weske, M.: Business process management: A survey. In: van der Aalst, W.M.P., ter Hofstede, A.H.M., Weske, M. (eds.) BPM 2003. LNCS, vol. 2678, pp. 1–12. Springer, Heidelberg (2003)
10. Ferreira, J.E., Takai, O.K., Braghetto, K.R., Pu, C.: Large scale order processing through navigation plan concept. In: IEEE SCC, pp. 297–300. IEEE Computer Society Press, Los Alamitos (2006)
11. Motta, E.: Reusable Components for Knowledge Modelling: Case Studies in Parametric Design Problem Solving. IOS Press, Amsterdam (1999)
12. The Business Rules Group: Defining business rules - what are they really? (July 2000), `http://www.businessrulesgroup.org`
13. Weske, M.: Business Process Management: Concepts, Languages, Architectures. Springer, Heidelberg (2007)
14. White, S.: Business process modeling notation (BPMN). Business Process Management Initiative (BPMI)–Version 1.0–BPMI.org (2004)
15. Workflow Management Coalition (WfMC): resource page (September 2008), `http://www.wfmc.org`