Generating Persistence Structures for the Integration of Data and Control Aspects in Business Process Monitoring

Eladio Domínguez[1], Beatriz Pérez[2], Ángel L. Rubio[2], María A. Zapata[1], Alberto Allué[3] and Antonio López[3]

[1]*Departamento de Informática e Ingeniería de Sistemas, Universidad de Zaragoza, Zaragoza, Spain*

[2]*Departamento de Matemáticas y Computación, Universidad de La Rioja, La Rioja, Spain*

[3]*Infozara Consultoría Informática, Zaragoza, Spain*

Keywords: Model–driven Approach, Development of Monitoring Systems, Process Flow and Data Integration, UML, Occurrence–centric.

Abstract: Today's organizations have to monitor increasingly complex business processes that handle large amounts of data. In this context, it is essential to design working frameworks that seamlessly integrate both control flow and data perspectives. Such an integration can be eased by automatically generating the infrastructures for storing data and control aspects. Towards this goal, we propose an automatic process for synthesizing persistence structures for control flow and data storage. In particular, based on an approach centered on the concept of *Occurrence*, in this paper we present a proposal by means of which, after applying several translation patterns to a business process model, we automatically generate the persistence structures that integrate both data and control aspects of such model. The feasibility of this proposal is demonstrated by developing a prototype and evaluating its application to different examples taken from the literature as a benchmark.

1 INTRODUCTION

Continuous improvement is an integral part of the life cycle of any business process. In order to achieve this objective, monitoring systems of process execution are required. From the information obtained through monitoring, it is possible to assess the adequacy of the processes to the reality and to proceed to the revision of the underlying models. Due to its intrinsic nature, business process management systems must be able to capture both the control flow perspective and the data perspective in order to faithfully achieve their goal. This is even more important when monitoring tasks are performed, in which the managed data must be aligned with the enacted processes. Often, these two perspectives have been treated separately, but more and more it is recognized (Eshuis and Van Gorp, 2016a; Eshuis and Van Gorp, 2016b; Kumaran et al., 2008; Künzle and Reichert, 2011; Nigam and Caswell, 2003) that a holistic and as much automated as possible approach of both aspects is required.

In (Domínguez et al., 2017; Domínguez et al., 2014) we presented an approach based on the notion of *Occurrence*, a three–dimensional modeling artifact that comprises guidance, structure and behavior (see Section 2). The final goal of that proposal is to improve the processes of design and development of monitoring systems, so that analysts and developers' work is lightened.

In this paper we propose a further step towards the development of monitoring systems that seamlessly integrate control flow and data aspects. We develop an improvement of the design strategy presented in (Domínguez et al., 2014), enriching it with automatic mechanisms for the construction of storage infrastructures for monitoring issues. This proposal has several potential advantages such as minimizing coding tasks and simplifying the management of the life cycle of business processes. In particular, in Section 3 we show how from a business process model, represented through a UML activity diagram that contains information about business objects (other business process modeling languages could also be used), a specific class model can be automatically generated. This class model, created by applying a UML profile and several patterns, comprehensively captures flow and data perspectives. Section 4 shows the details of

Domínguez, E., Pérez, B., Rubio, Á., Zapata, M., Allué, A. and López, A.
Generating Persistence Structures for the Integration of Data and Control Aspects in Business Process Monitoring.
DOI: 10.5220/0006781903200327
In *Proceedings of the 20th International Conference on Enterprise Information Systems (ICEIS 2018)*, pages 320-327
ISBN: 978-989-758-298-1
Copyright © 2019 by SCITEPRESS – Science and Technology Publications, Lda. All rights reserved

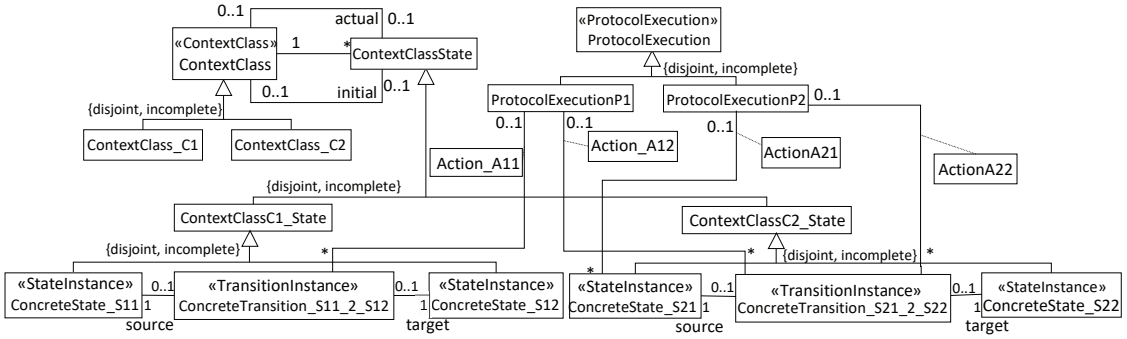

Figure 1: OcBase PIM pattern.

the translation and the automatization procedures of our approach, together with an implementation into an MDE–based tool, using the Atlas Transformation Language (ATL) (ATL Plug-in, 2017).

The proposal has been evaluated according to several criteria, such as performance and maintenance (Section 5). Other related works are discussed in Section 6, while Section 7 concludes with final remarks and future work.

2 BACKGROUND: THE OCCURRENCE– BASED APPROACH

Business process monitoring is concerned with recording information about the actual execution of processes in order to later extract valuable knowledge that can be utilized for business process quality improvement (Campanile et al., 2008; Kang et al., 2012; Kang et al., 2009; Reichert et al., 2010). The type of collected information influences the quality of the diagnosis accomplished during the monitoring phase. Trying to improve the diagnosis, in (Domínguez et al., 2017; Domínguez et al., 2014) a holistic perspective of system dynamics is proposed which enhances the monitoring of business processes as a whole. Specifically, when the dynamics of a system is defined as a set of processes accomplished by following several protocols, we suggested a three–dimensional business process monitoring approach, which simultaneously encompasses structure (objects), behavior (events, states, state changes) and guidance (protocols, processes and protocol performers).

According to this perspective, each protocol, when executed, acts on one or more objects, which react by changing their state and/or related data. An *Occurrence* is a concrete, identifiable and indivisible chunk of information which contains aspects organized according to three dimensions: guidance, structure, and behavior, given by protocol execution, object, and effect, respectively (Domínguez et al., 2014). The proposal also defines persistence structures that simplify the subsequent processing of the system trace. Occurrences, generated through protocol executions, are stored under a specific persistence structure, called *Occurrence Base (OcBase)*. Besides, an *Occurrence Management System (OcSystem)* is any system that enables the comprehensive management of occurrences, both in terms of their storage by using an OcBase, and in terms of their handling by using embedded tools that ease the exploitation of derived knowledge.

This proposal provides several theoretical concepts that can be applied using different methodological and/or technological approaches. In (Domínguez et al., 2014) we proposed a specific design strategy for the development of an OcSystem following an MDE approach (called *OcBaseMD design strategy*). In particular, this strategy provides several artifacts (the *Occurrence* profile and four patterns) for the design of an Ocbase, that is, for the design of the persistence structures that will store the occurrences generated through system execution. The main pattern provided by this strategy is the OcBase platform independent model (PIM) pattern (see Figure 1). This pattern makes use of the *Occurrence* profile which consists of four stereotypes corresponding to the executed protocols («ProtocolExecution»), acting on objects («ContextClass»), which react by changing their state («StateInstance» and «TransitionInstance»). Furthermore, this pattern establishes a skeleton of the OcBase class diagram in order to capture the hierarchical structure of state classes, and the transition classes among source and target state classes. Several advantages are obtained from using this design strategy that make it different from other monitoring proposals (Campanile et al., 2008; Kang et al., 2012; Kang et al., 2009; Reichert et al., 2010). Three of the most important advantages are: (1) a holistic perspective of system dynamics is provided, enhancing the monito-

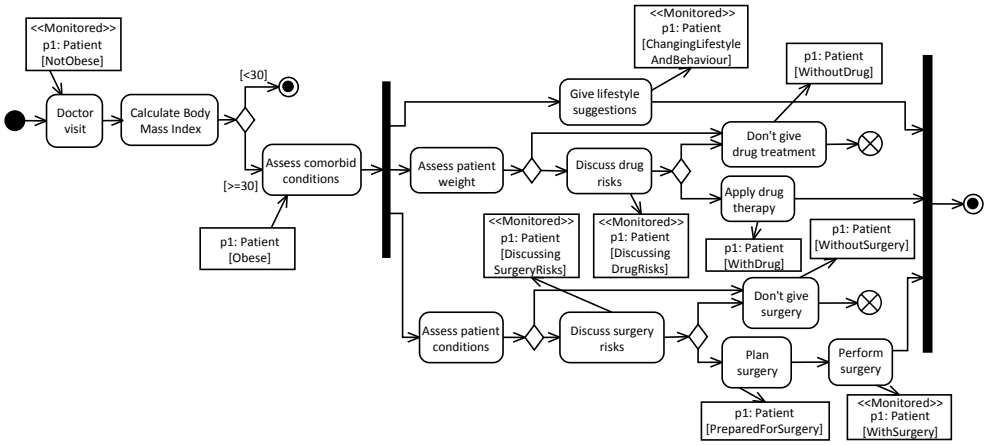

Figure 2: UML Activity Diagram of the obesity clinical guideline.

ring of business process as a whole (Domínguez et al., 2014), (2) an occurrence query framework is available which simplifies the querying process for obtaining provenance-related information (Domínguez et al., 2017), and (3) a single–pool solution for OcBase evolution is proposed for tackling the highly probable changes in the processes without incurring any lost of existing data (Domínguez et al., 2014).

In our research and development group (incorporating partners from both academia and industry), this approach has been applied to develop successful monitoring applications, such as a biobank OcSystem for identifying genetic and lifestyle aspects related to cardiovascular risk factors (Domínguez et al., 2017; Domínguez et al., 2014). Another example is the QRP Platform which is a Capability Maturity Model Integration (CMMI) appraisal tool for project quality management (Allué et al., 2013).

3 SYNTHESIZING PERSISTENCE STRUCTURES FOR CONTROL FLOW AND DATA STORAGE

We consider that, within the framework of a design strategy, the proposals for developing monitoring systems must include an automatic process for translating business process models into persistence structures. In this paper, we enrich the OcBaseMD design strategy presented in the previous section by providing it with a translation process that, starting from a process model, generates the storage infrastructure of an OcBase for storing the occurrences generated during the execution of that model.

Business process modeling can be performed using different conceptual modeling languages, such as BPMN (Object Management Group, 2011) or UML Activity Diagrams (Object Management Group, 2015), among others. The application of the previously presented occurrence–based approach requires a language that facilitates the representation not only of the control flow but also of the objects (structure) and their state changes (behavior) produced during the process execution. UML Activity Diagrams verify this requirement since it is a conceptual business process modeling language that allows us to specify the ordering of activities (control flow) as well as the flow of objects (data flow) that are used by the activities. In this paper, as an example, we assume that the business processes are represented by means of UML Activity Diagrams, but other languages would also be possible.

The control and data flows represented in a UML activity diagram are used for determining the type of elements that will be stored in the OcBase for monitoring purposes. With the goal of allowing the designer to take part in this decision, we have defined the *Monitoring* profile. This profile consists of a stereotype by means of which the designer, bearing in mind the monitoring requirements, can determine the objects and states that must be processed. The stereotype is called «Monitored» and extends the UML `ObjectNode` element. In this way, only the object nodes of the UML activity diagram with the «Monitored» stereotype will be taken into account during the generation of the OcBase schema. For example, Figure 2 shows a stereotyped UML activity diagram representing a real–life clinical guideline for the management of obesity in primary care (Snow et al., 2005). According to this guideline, different actions can be performed when a patient has a body mass index of 30 Kg/m^2 or greater. Note that several object nodes of type `Patient` appear in this activity diagram, but only some of them are monitored. We use this guideline as case study along the paper.

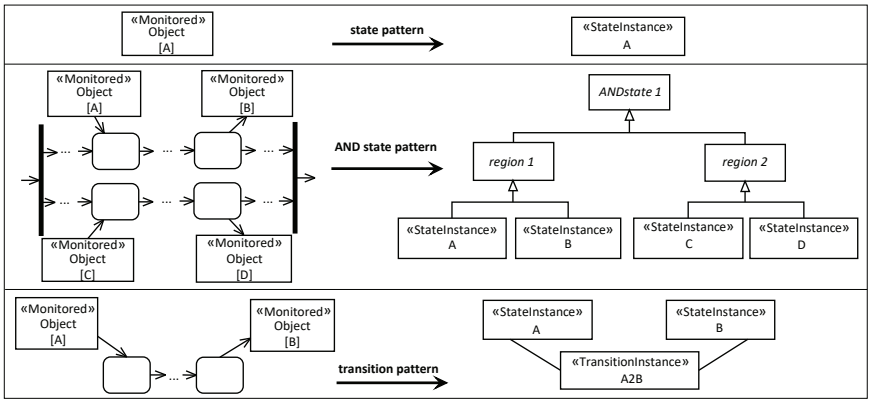

Figure 3: Translation patterns.

Figure 4: Translation process scheme.

The method for automatically generating the storage infrastructure of an OcBase (see Figure 4) takes as source models an activity diagram stereotyped with the *Monitoring* profile, and the *Occurrence* profile that will be applied to the target class diagram. Starting from these two models, the translation process synthesizes the UML class diagram of the OcBase, replicating the structure of the PIM pattern (see Figure 1).

The translation process takes the structure of the PIM pattern as scheme, so that several elements of the generated UML class diagram are common to all OcBases, independently of the designed UML activity diagram. Examples of these elements are the `ContextClass`, the `ContextClassState` and the `ProtocolExecution` classes. On the contrary, other elements of the OcBase, mainly the hierarchical structure of state classes, and the transition classes among source and target state classes, are determined depending on the given UML activity diagram. These activity diagram's dependent elements are created applying several translation patterns. Next, we explain three of the most outstanding patterns (see Figure 3).

State Pattern. For each monitored object node of the activity diagram, with state A, a class A representing the state is created in the OcBase.

AND State Pattern. This pattern is applied when there are stereotyped object nodes of the same object, entering or exiting activities which form part of a fork/join structure. In this case, a hierarchical structure of classes with one abstract class, and a region class for each flow of the fork, is created in the OcBase.

Transition Pattern. This pattern is applied when there is a pair of stereotyped object nodes of the same object in different states (A and B), one of them entering an activity and the other one exiting another activity (with no other object node of the same object in the intermediate activities). A class A2B representing the transition between the states is created in the OcBase.

4 AUTOMATIZATION OF THE TRANSLATION

In order to demonstrate the feasibility of our translation proposal, we have implemented it into an MDE–based tool, using the ATL Eclipse plug–in (ATL Plug-in, 2017), developing a first prototype. Specifically, the translation process has been implemented as an ATL module called `ActivityD2ClassD`, together with a simple ATL library called `UtilityLibrary`. The source models of these ATL units (that is, the stereotyped activity diagram and the *Occurrence* profile) must be provided in XMI syntax as `.uml` extension files and have to conform to the UML 2.5 metamodel. By using the defined units, the activity diagram is translated into an OcBase class diagram in XMI syntax, also conforming to the UML 2.5 metamodel. A complete explanation of our implementation of the translation process can be found in (Supplementary Material, 2018).

While the activity diagram could have monitored object nodes of any type, our prototype focuses on object nodes that refer to the same object (the translation of other design alternatives are discussed later in this section). Based on this, before performing the translation, the prototype must be provided with the information regarding the object and its corresponding type. Since ATL does not supply a way to parameterize variables' values at runtime, we include the corresponding pair of data *object-type* into an ATL helper defi-

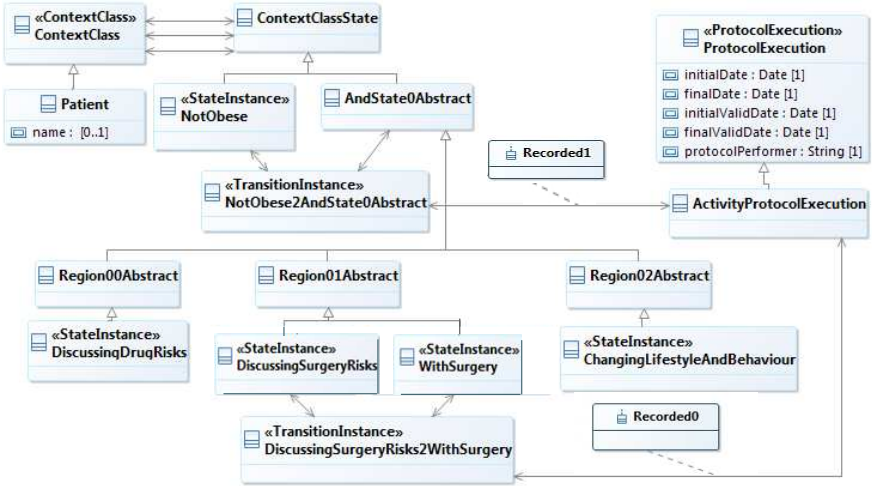

Figure 5: Generated Class diagram for the obesity case study.

ned within the independent `UtilityLibrary` library. In this way, not only do we separate such required information from the remainder translation, but also we anticipate to possible future improvements of the algorithm.

The ATL `ActivityD2ClassD` module, on the other hand, constitutes the pivotal part of the translation, implementing our patterns and using the information included in the `UtilityLibrary` library. The module consists of a set of 39 ATL helpers, which act as global variables and auxiliary methods, 1 matched rule and 17 called rules (around 1000 total lines of code). The strategy followed herein for implementing our translation patterns relies mainly on two of these rules: the `createPackage` and the `traceFromInitialNode`.

The called rule `traceFromInitialNode` constitutes the heart of the translation. The main aspect of this rule is that it is defined in a recursive manner, in such a way that in each call a new element in the source diagram is tackled. This rule performs two main tasks. Firstly, it traverses the overall activity diagram going through both the data and control flow, identifying the situations where our patterns can be applied. Secondly, as it traverses the activity diagram, it gathers and determines the information regarding the new elements in the class diagram to be created, which are activity diagram's dependent.

The second rule called `createPackage` is the only matched rule and starts the translation process. This rule is in charge of generating the overall stereotyped class diagram, including both the elements which are common to all OcBase class diagrams and those which are activity diagram's dependent elements. The first type of elements (that is `ContextClass`, the `ContexClassState` and the `ProtocolExecution` classes) are directly generated and the second type of elements are generated using the `traceFromInitialNode` rule.

The application of our prototype to the activity diagram of Figure 2 results in the stereotyped class diagram of Figure 5, which replicates the PIM pattern including, for example, the `Patient` context class, the `WithSurgery` state class or the transition class `DiscussingSurgeryRisks2WithSurgery`.

With regard to the scope of our proposed prototype, as described previously, it focuses on nodes that refer to the same object of a specific type. The prototype could be easily extended to deal with other design alternatives such as activity diagrams with different object nodes referencing objects o_1, o_2, etc. of a same object type t, or activity diagrams with multiple object types (t_1, t_2, etc.). In both situations the main issue corresponds to establishing the different hierarchy and source-target structures defined by the states of the different object nodes. In the first situation, we just need to register independently, per each object o_i, the structures defined by its object nodes through their different states. The second situation could be implemented similarly since the object nodes of a specific object type t_i would lead to independent parent-child and source-target structures. An implementation alternative could be to trace the diagram once per each object type and later join the results.

As for Activity diagrams' elements supported by our prototype, we would like to note that currently it does not support the existence of action nodes with several outgoing control flows, loops, cross-synchronization, or fork nodes with more than one associated join node, but alternative implementations could be considered. An interesting issue for future work would be to tackle the automatic mana-

Table 1: Case studies considered.

id	Case study	# Action nodes	# Object nodes	# Forks/ joins	# Decision/ merges	# Control flows	# Object flows
CS1	Obesity	13	10	2	5	28	10
CS2	Bikeshop	16	11	6	2	28	15
CS3	Dermatology	17	5	4	10	39	5
CS4	Create catalogue	10	6	2	2	13	11

gement by means of the prototype of synchronization structures where each fork node could have more than one associated join node, as well as of cross-synchronization.

5 EVALUATION OF THE ATL-BASED PROTOTYPE

This section investigates the strengths and weaknesses of our prototype. More specifically, we have applied it to four different case studies and we have analysed the results in the light of several criteria regarding: (1) the amount of class diagram elements that were generated, (2) the time it took to generate the class diagrams, and (3) maintenance aspects.

Table 1 shows the four case studies we have considered indicating the number of elements of the source activity diagrams. The first case study (CS1) corresponds to the obesity clinical guideline used as case study along the paper. In this case, we have used the UML 2 Eclipse plug-in to obtain the `.uml` extension file. The other three examples were determined starting from the online example models proposed in (Eshuis and Van Gorp, 2016b) (they were available in `.xmi` or in `.uml` extension format). The case studies we have selected correspond to: (CS2) the order and delivery of bikes, (CS3) the handling of dermatology patients and (CS4) the creation of a catalogue. We have carried out several changes to these three examples to be able to apply our proposal (that is, eliminating loops and forks with more than one associated join).

We have applied our prototype to these four case studies using a personal computer, Intel(R) *Core*^{TM} i5 CPU, 3.2 GHz, with 4 GB RAM, running Windows 7 Enterprise. We have performed two types of experiments for each case study: (1) monitoring all the object nodes in the model and (2) monitoring around half of the object nodes in the model. Each type of experiment has been repeated five times per case study, measuring the response times to produce an average time for each type of experiment and case study.

Table 2 shows the results obtained from our experiments, ordering them according to the number of monitored object nodes. It can be seen that, in each case (all the nodes and half of the nodes), the greater the number of monitored objects the greater the response time. Besides, all the response times are less than half a second, with a very small difference among them. These values, as well as the relation between the response time and the number of monitored objects, can be seen as a good performance result.

As for maintenance aspects, our ATL-based prototype would face with the evolution of business process models better than a manual not pattern-based approach. Specifically, every activity diagrams' design change can be automatically processed without software designers' interaction.

An Eclipse distribution with the ATL transformations, together with relevant documentation regarding the benchmark of case studies used for the evaluation are available from (Supplementary Material, 2018). We encourage the interested reader to try it out.

6 RELATED WORK

The need of considering data (both managed and generated) as *first class citizens* within business process modeling has been previously tackled in the literature. One of the first works that exposes this need is (Nigam and Caswell, 2003), which gives the notion of *business artifacts* as "information chunks that the business creates and maintains." This concept is related to the Occurrence notion, although this construct encompasses data and behavior.

Another similar approach is that of (Kumaran et al., 2008), which defines *business entity* as a "dominant information entity with an associated data model and an associated behavior model in the context of a process scope". Although this proposal shares one of its main objectives with our approach, it is very specific since it stands up for modeling the behavior of business entities with state machines. The Occurrence approach is more general and does not restrict the use of any specific technique for information representation.

There exist other papers, such as (Hull et al., 2010) or (Künzle and Reichert, 2011), that make innovative proposals for the comprehensive management of objects and behavior. However, these works move away from the most common standards in the domain. Our approach is demonstrated by its adaptation to any standard (in this paper, as an example of application, UML models are used). An example of a work that is based on a standard is (Friedenstab et al., 2012), which presents a BPMN extension to improve monitoring. We note that this work also uses the term occurrence (but in a sense very different from ours)

Table 2: Time results obtained for the case studies.

id	Case study	monitoring all the object nodes				monitoring half of the object nodes			
		# Monitored object nodes	# State classes	# Transition classes	Translation average time (s)	# Monitored object nodes	# State classes	# Transition classes	Translation average time (s)
CS2	Bikeshop	11	10	10	0.327	6	5	4	0.275
CS1	Obesity	10	10	7	0.308	5	5	2	0.273
CS4	Create catalogue	6	4	2	0.261	3	3	2	0.248
CS3	Dermatology	5	5	5	0.252	2	2	1	0.243

but it does not deal with automation aspects. In (Herzberg et al., 2013) an approach to use object state transition events for reasoning about process progress is given. In that proposal, very simple state machines are considered, while in our approach we face the difficulties that come from considering parallel flows and concurrent states.

Several papers propose model transformations for business process management. Eshuis and Van Gorp (Eshuis and Van Gorp, 2016a; Eshuis and Van Gorp, 2016b) automate the generation of object life cycles (UML state machines) from business process models (UML activity diagrams). Our proposal goes a step further, since it allows the generation of class diagrams that include intrinsically and comprehensively both data and flow perspectives. Other works (Brđanin and Maric, 2012; Rodríguez et al., 2010), closer to our proposal, automatically transform UML activity diagrams into UML class diagrams. However, in (Brđanin and Maric, 2012) authors do not consider alternative (decision/merge) and concurrent control flows (fork/join), and the work presented in (Rodríguez et al., 2010) pays special attention to security requirements. As far as we know, our automatic transformation from activity diagrams to class models is the only one proposed within the process monitoring context.

7 CONCLUSIONS AND FURTHER WORK

We have presented an approach to automatically generate storage infrastructures for monitoring tasks that integrates control flow and data perspectives. We have shown the feasibility of the proposal using UML Activity Diagrams, as business process modeling technique, and an MDE approach, as a design strategy. After applying our approach to different case studies, we can conclude that the results obtained from our evaluation are promising. In particular, the advantages of our approach could be summarized as minimizing coding tasks, as well as simplifying the management of the life cycle of business processes.

There are several lines of further work. At present, as said previously, there are specific structural situations not tackled by the prototype. These situations deserve to be investigated. Furthermore, a special remark must be made regarding the *pin–style* modeling of object flows. Although the basic pin–style is equivalent to the object node style (Object Management Group, 2015), the general pin–style notation is more expressive allowing to handle more complex situations such as alternative pin sets. Thus, considering the general pin–style notation is an issue of future work.

ACKNOWLEDGEMENTS

This work has been partially supported by the spanish Ministry of Economy and Competitiveness (project EDU2016-79838-P) and the University of Zaragoza (project UZ2015-TEC-05).

REFERENCES

Allué, A., Domínguez, E., López, A., and Zapata, M. A. (2013). QRP: a CMMI Appraisal Tool for Project Quality Management. *Procedia Technology*, 9:664–669.

ATL Plug-in (2017). http://www.eclipse.org/atl/. Last visited on January 2018.

Brđanin, D. and Maric, S. (2012). An approach to automated conceptual database design based on the iml activity diagram. *Comput Sci Inf Syst*, 9(1):249–283.

Campanile, F., Coppolino, L., Giordano, S., and Romano, L. (2008). A business process monitor for a mobile phone recharging system. *J. Syst. Archit.*, 54(9):843–848.

Domínguez, E., Pérez, B., Rubio, A. L., Zapata, M. A., Allué, A., and López, A. (2017). Developing provenance-aware query systems: an occurrence-centric approach. *Knowl Inf Syst.*, 50(2):661–688.

Domínguez, E., Pérez, B., Rubio, A. L., Zapata, M. A., Lavilla, J., and Allué, A. (2014). Occurrence-Oriented Design Strategy for Developing Business Process Monitoring Systems. *IEEE Trans. Knowl. Data Eng.*, 26(7):1749–1762.

Eshuis, R. and Van Gorp, P. (2016a). Synthesizing data-centric models from business process models. *Computing*, 98(4):345–373.

Eshuis, R. and Van Gorp, P. (2016b). Synthesizing object life cycles from business process models. *Software & Systems Modeling*, 15(1):281–302.

Friedenstab, J.-P., Janiesch, C., Matzner, M., and Muller, O. (2012). Extending bpmn for business activity monitoring. In *Proc. of HICSS'12*, pages 4158–4167. IEEE.

Herzberg, N., Meyer, A., Khovalko, O., and Weske, M. (2013). Improving business process intelligence with object state transition events. In *Proc. of ER'13*, volume 8217 of *LNCS*, pages 146–160. Springer.

Hull, R., Damaggio, E., Fournier, F., Gupta, M., et al. (2010). Introducing the guard-stage-milestone approach for specifying business entity lifecycles. In *Proc. of WS-FM'10*, volume 6551 of *LNCS*, pages 1–24. Springer.

Kang, B., Kim, D., and Kang, S.-H. (2012). Real-time business process monitoring method for prediction of abnormal termination using KNNI-based LOF prediction. *Expert Systems with Applications*, 39(5):6061–6068.

Kang, D., Lee, S., Kim, K., and Lee, J. Y. (2009). An OWL-based semantic business process monitoring framework. *Expert Systems with Applications*, 36(4):7576–7580.

Kumaran, S., Liu, R., and Wu, F. Y. (2008). On the duality of information-centric and activity-centric models of business processes. volume 5074 of *LNCS*, pages 32–47. Springer.

Künzle, V. and Reichert, M. (2011). Philharmonicflows: towards a framework for object-aware process management. *J. Softw. Maint. Evol.: Res. Pract.*, 23(4):205–244.

Nigam, A. and Caswell, N. (2003). Business artifacts: An approach to operational specification. *IBM Syst. J.*, 42(3):428–445.

Object Management Group (2011). Business Process Model and Notation v2.0 formal/2011-01-03. http://www.omg.org/spec/BPMN/2.0/PDF. Last visited on January 2018.

Object Management Group (2015). Unified Modeling Language v2.5 formal/2015-03-01. http://www.omg.org/spec/UML/2.5/PDF. Last visited on January 2018.

Reichert, M., Bassil, S., Bobrik, R., and Bauer, T. (2010). The proviado access control model for business process monitoring components. *EMISA*, 5(3):64–88.

Rodríguez, A., de Guzmán, I. G.-R., Fernández-Medina, E., and Piattini, M. (2010). Semi-formal transformation of secure business processes into analysis class and use case models: An mda approach. *Inform Software Tech*, 52(9):945–971.

Snow, V., Barry, P., Fitterman, N., Qaseem, A., and Weiss, K. (2005). Pharmacologic and surgical management of obesity in primary care: a clinical practice guideline from the American College of Physicians. *Ann Intern Med*, 142(7):525–31.

Supplementary Material (2018). Generating persistence structures for the integration of data and control aspects in business process monitoring. Available at: https://zenodo.org/record/807684#.WbqoAPMjG70. January 2018.

SCITEPRESS SCIENCE AND TECHNOLOGY PUBLICATIONS