A new approach for workflow evolution using MDA technology

Berraouna Abdelkader*

Department of Information Technology,
University Badji Mokhtar-Annaba, Algeria
and
University of Chérif Messaadia Souk Ahras,
BP 12, 23000, Annaba, Algeria
Email: Abdelkader.br@gmail.com
*Corresponding author

Amirat Abdelkrim

Computer Science Department,
Mohamed Cherif Messaadia University, Souk Ahras,
BP 1553, Souk Ahras, 41000, Algeria
Email: abdelkrim.amirat@yahoo.com

Meslati Djamel

LISCO Laboratory,
Computer Science Department,
Badji Mokhtar University,
BP 12, 23000, Annaba, Algeria
Email: meslati_djamel@yahoo.com

Abstract: Most companies, independently of their sizes and activities types, do not provide sufficient adaptability and evolution of their workflow needed to deal with changes in business agreements and organisation methods. In this paper, we treat the adaptability and evolution of workflow based on MDA technology. And we present our new approach for workflow evolution at the design-time, in this approach, the workflow model is described by a specification at a very high level of abstraction using meta-model concepts. The workflow model can be evolved by the workflow designer using model evolution operations. The evolution of models can be managed in different ways. The designers can refine the workflow model according to the evolution scenario proposed by the company manager. In this work, we have developed endogenous rules in ATL which allow the transformation and evolution of workflow model, and among these rules, we have used the endogenous transformation model that affects models expressed in the same language.

Keywords: meta-model; workflow; e-commerce business; dynamic evolution MDA; flexibility; Atlas Transformation Language; ATL.

Reference to this paper should be made as follows: Abdelkader, B., Abdelkrim, A. and Djamel, M. (2019) 'A new approach for workflow evolution using MDA technology', *Int. J. Intelligent Information and Database Systems*, Vol. 12, No. 4, pp.304–334.

Biographical notes: Berraouna Abdelkader is an Assistant Professor in the Department of Computer Science at University of Souk-Ahras, Algeria and Visiting Researcher at IRISA Laboratory of Bretagne Sud University Vannes, France. He is a PhD student at the University of Badji Mokhtar-Annaba, Algeria. He received his Magister in the Department of Information Technology at same university, in 2014. He has published several papers on the business processes modelling and workflow evolution in several international conferences. His current research interests include workflow evolution, MDA technology and business process development.

Amirat Abdelkrim received his PhD in Computer Science in 2007 and Habilitation in 2010. Currently, he is a Full Professor of Computer Science at the University of Souk-Ahras, Algeria. He is the Director of the Mathematics and Computer Science Laboratory and Chief of Software Engineering Team. His main research concern is software architectures and their evolution, modelling and meta-modelling. He worked on several national projects in software engineering. He has published several refereed journal and conference papers in the fields of software architecture, component-based and object-oriented modelling. He has served on program committees of several international journals, conferences and workshops.

Meslati Djamel is a Full Professor in the Computer Science Department of Badji Mokhtar University, Annaba, Algeria. He heads the LISCO Laboratory. He published several papers in international conferences and journals. He has been a program committee member of several conferences. His research interests include bio-inspired systems and software engineering.

1 Introduction

Companies are always changing and want to adapt the environment around them, to stay relevant, innovative and competitive. These companies always need to improve their workflows to gain more time and more money. Workflow is used to describe the automatic task execution cycle time, validation methods and to provide each user the information necessary for the execution of his task and most have the same automation with small differences in its characteristics (Workflow Management Coalition Glossary, 1996). The design and implementation of these workflows require much study, time and effort. Nowadays, the market situation is dynamic; various conditions are created by changes in the organisation's market environment (Ngai et al., 2005; Hehenberger et al., 2016; Kiu and Lee, 2017). In this context, adaptation and dynamic workflow evolution can be considered as the main objective of company manager, both at the organisational and operational levels, adaptation refers to the ability to effectively adjust under various conditions.

This work is motivated by the lack of methods for the adaptation and the dynamic evolution of workflow systems. Different approaches were found in the literature. However, these approaches with theirs features do not provide sufficient flexibility needed to deal with certain situations that may occur during the execution of the workflow and does not have a global vision of workflow evolution. Also, these approaches present different solutions that are very specific to their respective domains and they are difficult to be reused in other domains.

In our research, we have investigated multiple workflows of several commercial enterprises, in order to create a general meta-model to start or create an instance (model) of this meta-model and help these new companies to establish their own workflow. And giving the opportunity for evolution using Atlas Transformation Language (ATL) rules (Kalyanasundaram and Ugale, 2015) to allow making changes in all aspects of workflow. 'The evolution in our approach is considered as model transformation operations implemented using the transformation language ATL'. Workflow needs to adjust effectively to changes. Since items that organisations give to the market are created by business forms, adaptable workflow is an essential resource for adapting to market changes in a compelling way.

Distinctive viewpoints must be considered while considering adaptability. As a matter of first importance, adaptability is given by express representation of workflow, since adjustments of unequivocal, graphically indicated workflow is much less demanding than adjustment of composed authoritative methodology or business approaches covered in programming level. Specifically, based on a workflow meta-model, we propose a set of elements changes operations and designer intervention operations. Also, examine how these can be utilised in order to support the controlled execution of potentially incompletely specified and dynamically changing. The business of companies is changing little in relation to the technologies they used to build their IT applications (Sohail et al., 2016). For this reason, it is good practice to separate commercial and technical aspects when developing a workflow system.

This makes it possible if the business specifications premed and independently of the technical specifications. The most important advantage of modern driven architecture (MDA) is the sustainability of specifications. It is always important for a company to increase its productivity in order to be competitive. One well-known way to increase productivity is to automate certain production steps. This is what MDA does by automating model transformations (Lengyel et al., 2008; Vara et al., 2010).

Our approach is based on a process meta-model which serves as a frame for the method of development approach that takes into account the evolution and flexibility of these systems.

The method is defined around three main phases and has MDA (Kim et al., 2016) orientation since it distinguishes platform independent model (PIM) and platform specific model (PSM) and it is based on the process meta-model. In addition, the use of MDA in our approach allows us to describe the system components and their functions independently of their implementation and to promote the reuse of its components.

When change occurs, the process model can be evolved or extended in one way or another, rather than rebuilding a new model. The evolution may be considered or must be applied on a temporary or permanent basis. This evolution takes place at two levels:

1 in the workflow model (in design-time)

2 in instances running (in run-time).

Our proposed approach is implemented at design-time, while other approaches are implemented for run-time (Dohring and Zimmermann, 2011; Milanovic et al., 2011).

The design-time adaptation has certain properties that may make it advantageous in certain respects. Run-time approaches create variations model on concrete instances at run-time. In this sense, they are complementary with design-time approaches.

The workflow model can be evolved by the workflow designer by applying model evolution operations. The evolution of models can be managed in different ways. An evolution strategy implements only one of the many possible adaptations. The designer can refine the workflow model according to the evolution scenario proposed by the company manager.

In other words, the refinement of models creates different adaptations, depending on the type of evolved model; each adaptation is saved in the library. Adaptations can be customised to meet the different evolution scenarios.

A model evolution is a modification of a property of an existing model concept. The body of a rule can introduce arbitrary adaptations into a model by making assignments to model properties.

We have four types of evolution operations:

- adding a concept
- duplication of concepts
- editing a concept
- deleting a concept.

Our contribution could be summarised as follows:

- In our approach, the workflow model is described by a specification at a very high level of abstraction using meta-model concept, this specification is provided by a workflow meta-model that captures different aspects of workflows and this meta-model includes concepts to capture the functional, structural, informational and behavioural flows of workflows.
- We have created a rule library to make the evolution, this evolution is applied based on changes made to the model and can be extended or even modified by designers, who can specify customised evolution rules to refine the model. In this library, we have used the endogenous transformation model that affects models expressed in the same language. In this transformation, we have focused on refinement of high-level workflow model.
- In our approach, we use small commands that automate repetitive tasks in the model transformation, because manual model evolution can generate errors and inconsistencies in the model.

Technically, we have used ATL rules to implement this evolution by creating a library of endogenous rules that allows the evolution of the workflow and gives the possibility to evolve in all aspects of the workflow.

To illustrate our system, firstly, we present a meta-model of workflows in e-commerce business domain, after we present a scenario dealing with an example of a commercial workflow and modelled authority from this meta-model, we will make an example of the evolution of this workflow in ATL. In particular, based on an activity meta-model, we propose a set of dynamic change operations and user intervention operations.

The rest of this article is organised as follows: Section 2 presents some of basic concepts and terminology of dynamic workflow and context of our research work. Section 3 outlines the similar work and highlights the motivation of this work. Section 4 presents our proposed approach. Section 5 presents a case study and Section 6 will provide conclusions and some perspectives.

2 Concepts and terminology of dynamic workflow

Before presenting the existing approaches for the modelling of adaptive workflow, we first present several related concepts, in order to make this paper self-sufficient.

2.1 Workflow

A workflow is a software tool dedicated to the management tool processes. This tool defines, manages and executes processes, by implementing programs in which execution order is pre-defined in a computer representation of the logic of these procedures (Workflow Management Coalition Glossary, 1996).

2.2 Workflow evolution

Workflow evolution concerns the permanent modification of workflow models to obtain a different version. Workflow evolution is a real problem, as it is regularly associated with changes in business arrangements and business process organisation methods not just with the problem of changing task aggregations and their accuracy.

Thus, issues identified with the redesign, version control and substitution of workflow models must be considered in accordance with business process enhancements. Ad hoc adaptation of workflow models happens dynamically as the workflow models are being executed (Han et al., 1998).

2.3 Adaptability

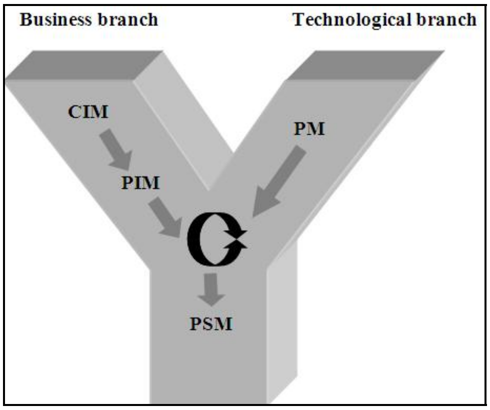
Adaptability is the process capability to deal with exceptional cases and a non-standard behaviour. This can be partially solved in time by adjusting the design (adapter accurately) the structure of the process model (Kalyanasundaram and Ugale, 2015; Yang and Xu, 2011; Suganthi and Nadarajan, 2013).

2.4 Model driven architecture

The MDA is a proposed development approach and supported by the Object Management Group (OMG) (Kalyanasundaram and Ugale, 2015). The main idea of MDA is to distinguish the functional specifications of a technical specification system from its implementation on a specific platform. In other words, this approach allows the same model to be used on multiple platforms using projections. The MDA approach is entirely based on models and their transformations.

The MDA development cycle is seen as a Y (see Figure 1) where the branches represent the functional and technical specifications of the target platform, once integrated, lead to implementation. Therefore, the difference between the model and the system does not exist anymore, because the model is the system, or at least is supposed to be automatically and directly generated from the model.

Figure 1 The MDA approach development cycle



The main level of abstraction corresponds to the computation independent model (CIM). The CIM plans to comprehend the mission of the considered framework as depicted in domain analysis step. The second level of abstraction is the PIM. This model spotlights on describing the system components and their functions independently of the implementation platform. The last level is the PSM. The PSM intends to determine a model which can be utilised to produce code particular for a platform.

2.5 ATL

The ATL is the Atlas Research Group answer to the OMG MOF/QVT RFP. It is a model transformation language specified both as a meta-model and as a textual concrete syntax. ATL is made available as part of the ATL development tools (ADT) (Jouault and Kurtev, 2006).

3 Related works

Currently, there is a wide range of techniques have been proposed for workflow evolution modelling in different application domains, such as:

- rule-based approaches
- constraint-based approaches
- case-based approaches
- aspect-based approaches
- meta-modelling-based approaches

- web service composition
- grid computing
- software management
- architectural reconfiguration.

In this paper, we have studied several approaches for modelling dynamic workflow. For example, the Yet Another Workflow Language (YAWL) system (ter Hofstede et al., 2010) gives support to adaptability and dynamic exception and uses the concept of worklets, extensible directory of autonomous sub-processes with related determination rules. This manages complex data transformations and complete integration with organisational resources and remote web services, YAWL offers:

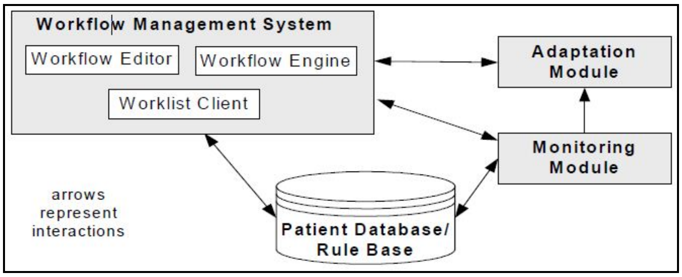
- Strong process specification language to identify control-flow relationships and resource requirements. Native data processing through XML schema, XPath and XQuery.
- A formal basis that allows its specifications to be unambiguous and allows automated checking.
- A service-oriented architecture that offers an infrastructure which can be easily adapted to specific requirements. This approach gives a dynamic change and advancement forms without intercession outside the framework on the other hand stop (ter Hofstede et al., 2010; Adams et al., 2015).

Case-based reasoning (CBR) approach: for the automated adjustment of the workflow through the reuse of the experience. This approach facilitates the execution of adjustments by an automated method. The method utilises the workflow adjustment of cases recorded in past adjustments. The recorded modifications can therefore, be exchanged according to a segmented approach, the new work process which is in a comparable situation, is transformed using the CFCN (workflow modelling language developed by the University of Trier).

ADEPT2 model perform the change procedure at run-time (i.e., include, erase and alter the grouping of errands) to the model level (element advancement) and at instance (ad hoc changes). These alterations are made to a conventional homogeneous model and should be accomplished through manual mediation of an administrator, outlined in an abnormal state connection. The framework is additionally in charge of the ‘jump forward’ and ‘backward jumps’ in the process instances, not just by the approved administrator (Mejri et al., 2015).

Greiner et al. (2005) proposed a prototype ‘AdaptFlow’ workflow system that provides support in the field of oncology therapies by combining a workflow management system with an event-condition-action (ECA). The dynamic adaptation of the workflow is done according to one of the two ‘predictive’ or ‘reactive’ strategies: AdaptFlow tries to adapt in advance the remaining part of a current therapy workflow as soon as an exception is detected. This informs users at an early stage of the necessary changes and promotes effective patient treatment. This strategy is based on time estimates of the duration of future parts of the workflow, which are not always possible. In these cases, the second strategy reactive adaptation is used which adapts directly affected activities before their execution.

Figure 2 'AdaptFlow' workflow systems


Workflow Management System
Workflow Editor
Workflow Engine
Worklist Client
Adaptation Module
Monitoring Module
Patient Database/ Rule Base
arrows represent interactions

The literature provides several meta-model for business process modelling used for conceptual design to define the elements of the workflow language and their interrelationships. The meta-model approach is utilised (expressly or verifiably) to identify the structures and types of workflow model from the constituent components. An arrangement of primitives is ordinarily characterised in which change operations can be played out a certain model or bodies. Examples of this approach incorporate ADEPTflex (Allehyani and Reiff-Marganiec, 2016; Reichert and Dadam, 1997), WASA (Weske, 1998) and WIDE (Casati et al., 1996).

A meta-model describes the structure of models and permits reasoning on models like the knowledge of first level. Some programming languages, like Java, depict the same way the classes, meta-classes and objects. In programming, it is with the idea of the UML meta-model that has been created. The meta-model describes the models: classes, objects, characteristics, connections and self-described, additionally implies the model of the model. It can be characterised as the representation of a specific view on model. The syntactic precision and conformities can be made by customers. In MDA approach, every workflow depends on a procedure model that has been improved with extra attributes that permit its implementation. Workflows models are described utilising coordinated diagrams of which bunch represent activities. From the perspective of the workflow, management system basic capacities are optimised.

The workflow management system has an outside view on the automated or manual execution of the capacities, however, is not worried with the execution of the capacities themselves.

- The object system represents the subjectivity of this present reality, including the pertinent part of the environment.
- The model system represents to the subjective picture of the object system. It utilises a sentence structure to make the model system.
- The projection details the relationship between the object system and the model system.

If a model system M1 represents to the object system of a model system M2, then the model system M2 represents the meta-model configuration of the object system M1. Because of this high level of abstraction, a meta-model can be seen as a schema structure that identifies the basic components of the model and the links between the components of the model and their semantics.

In WorkParty (Siemens Nixdorf Informationssysteme AG, 1995), a process is a coordinated graph that decides the succession of activities or projects respectively. It comprises of a beginning activity and at least one ending activity, connectors and knots. The beginning activity contains information about the process that is important for all activities. Knots contain exercises or projects and connectors decide their sequence. Some final activities can be used to model the option ends of a process and the activity at the beginning must be unique. An activity is a basic step in a process. Activities can be manual or automatic. Besides, programs might be connected with activities that are just executed if there should arise an occurrence of an exception. WorkParty also has ad hoc activities; at run-time the client may insert a boundless number of new activities at a predetermined point, before the process is continued.

The WorkParty uses a database to store process and activity models. A subject document includes all process and activity models that are identified with a particular business area. A model file is a subset of a subject document and contains process and action layouts for a specific sort of business process. Handle records are instances of model files and contain connections and parameters for the particular processes.

IBM's work process administration framework FlowMark (IBM, 1996; Fraj et al., 2017) use process graphics to characterise workflow process models. In FlowMark, each work process model is indicated by a coordinated graph, of which the concentrators talk about activity and the edges are distributed in an arrangement of control flow edges and an arrangement of data flow edges. Control stream edges speak to potential control stream, characterised by translating conditions, which are predicates evaluated at run-time. An activity may have written info, output parameters and data flow interface parameters of various activities. FlowMark depends on a detachment of a workflow process built time and its run-time. Workflow is modelled in built time and executed in run-time.

More specifically, workflow models cannot be modified after the build time. In this way, the enhancement of adaptability is quite limited, dynamic changes to workflow models are not accepted by FlowMark. In addition, customers cannot change the control flow of dynamic workflow instances by ceasing or skipping certain activities. Since this is preferably a constraint of the FlowMark framework rather than the display language of the work process and since the graphical representations of workflow models are natural, FlowMark and WorkParty offer the customer a graphical demonstration tool for defining workflow models.

In the FlowMark work process, models are drawn as diagrams with one-sided transitions over the express design of knots and connectors. WorkParty offers the client with predefined process components, to give control flow components, for example, forks and joins.

Leitner et al. (2011) handled the security issue in adaptive workflow frameworks and proposed a role-based access model to adapt to the auto-adaptive workflow.

Liu et al. (2012) proposed a procedure for continuous improvements of workflow execution through a productive resource designation. For this reason, they exploit workflow execution logs and rule a mining algorithm to extract new knowledge about tasks and resources binding.

Muller et al. (2004) proposed an agent-based framework for automatic workflow adaptation, when undesirable events occur. This arrangement measures the execution time of the business process to be chosen when workflow adaptations are required.

El Fazziki et al. (2014) proposed an agent oriented model driven approach to fill the gap between systems development and business requirements. They take benefit of the agent's adaptability and learning ability to ensure an autonomous adaptation of the workflow management system to cope with changes using the Q-learning algorithm.

The implementation of this framework is mainly based on the goal process modelling notation (GPMN), the agent modelling language (AML), and the agent platform JADE. To cope with the need of enhancing the adaptability workflow systems, many solutions have been proposed.

In the system structuring step, they use the MDA to specify the agents composing the system. The MDA consists on describing system as different models expressed with various levels of abstraction.

In their work, they concentrated on exploiting the agent paradigm and the objective arranged view to overcome any issues between work process administration framework and adaptability. They propose an organised approach in view of MDA to fabricate a multi-operator framework. The MDA enabled us to extricate operator from the GPMN chart speaking to the objective arranged perspective of BP. This was accomplished through GPMN/AML mapping chopped out for the most part on partner a specialist to each distinguished.

Ben Fraj et al. (2017) proposed a meta-model transformation for the detail and the execution of cloud service adaptable workflows. To built workflow conceptual models, the proposed approach utilises a BPMN model for the specification of the cloud service workflow structure and the state-chart diagram for the determination of the cloud service workflow behaviour. Workflow models should be converted to the BPEL4WS language which will be performed by the BPEL4WS engine. The last is driven by the conduct portrayed by the state-chart diagram. To satisfy, they characterise an arrangement of meta-model transformations from the PIM (BPMN) to the PSM (BPEL4WS).

They proposed a constant way to built with fabricated cloud administrations applications by following OMG(s) principals of the MDA in the improvement procedure. In this approach, the workflow modelling recognises the asset starting with one portrayed cloud administration's task then onto the next to assemble and creates the entire application. To model and express the made adaptable workflow out of cloud administrations, they characterised an arrangement of steps which start from the detail of the cloud benefit work process structure with BPMN and wrap up by preparing the adaptable properties of the work process display utilising the real time meta-model transformation.

Hamri et al. (2007) have proposed an approach utilising the MDA engineering, to build a workflow meta-model utilising meta-object facility (MOF) and to exploit ontologies for defining the concepts that portray the information of the workflow domain. This supposed combination of methodologies has been used in many research projects. This design is a MDA defined ontology engineering that incorporates an ontology definition meta-model. This approach is in accordance the MDA rule in view of interpretations between PIM and PSM for generating the code. The basic meta-model and ODM are then treated as two PIMs and the OWL meta-demonstration is the PSM.

Popp et al. (2014) proposed to demonstrate high-level reference processes with less detail than the completely-fledged procedures. The missing points of interest are caught in extra business decisions that implement how an association performs random errands. Their application to an abnormal state reference process prompts its refinement. In fact, they propose an express division of process models (represented in BPMN) and particulars of business rules applying such guidelines as model changes to process models (at configuration time) prompts models of either refined or balanced models of business forms. Thus, even with no expansion of the BPMN 2.0 standard, these model adjustments can deal with the required changes of business processes.

Tantan and Akoka (2014) presented a novel approach for transforming business rules communicated with the semantics of business terminology and rules into business process models. This transformation gives a few advantages to workflow system, for example, upgrading prerequisite approval and refinement, enhancing business processes documentation and decreasing their general demonstrating exertion.

To conquer any obstruction amongst SBVR and BPMN, they proposed a novel transformation decision that cover basic and conduct SBVR rules.

Rhazali et al. (2016) proposed a strategy which checks the transformation of the model from CIM level to PIM level conform to MDA approach. Their approach is established on building up a decent CIM level, through all around selected rules, to encourage change to PIM level. In any case, they created a high level of PIM using use case diagram models, state diagram models, class diagram models and package diagrams. At that point, build-up establish transformation rules to guarantee a semi-automatic transformation from CIM level to PIM level. Their approach offers a solution to the problem of transformation from plans of action represented on CIM level to configuration models represented to on PIM level.

These approaches presented different solutions that are very specific to their respective domains. In addition, they are difficult to reuse in other areas and in some cases, in other applications in the same area.

However, these examined approaches with theirs features do not provide sufficient flexibility needed to deal with certain situations that may occur during the execution of workflow and does not have a global vision of workflow evolution.

Compared to existing approaches, the main contributions of our approach are:

- We proposed a generic transformation approach that is not directed to a particular domain and the creation of a general meta-model, which captures different aspects of workflows, this meta-model includes concepts to capture the functional, structural, informational and behavioural flows of workflows and allows its specification at a very high level of abstraction.
- We have created a library of endogenous rules in ATL for dynamic workflow evolution. This library allows endogenous transformation model that affects models expressed in the same language. The two source and target models are consistent with the same meta-model.
- Giving the opportunity for workflow evolution using ATL rules to allow making changes in all aspects of workflow.
- Based on a workflow meta-model, we propose a set of elements changing operations and designer intervention operations.
- We have focused on refinement of high-level workflow model.

In addition, compared to other approaches which use meta-model, our workflow meta-model has the following distinctive features:

- On the basis of the workflow meta-model, the problem of the evolution of the model has been studied to allow this evolution.
- For the workflow meta-model, a set of evolution operations has been proposed which allow the model to evolve in different ways. This set includes the operations of adding and deleting workflow concepts as well as modification and duplication operations.
- The workflow meta-model explicitly supports the evolution of workflow models. The workflow model can be evolved by creating a new version or by modifying it using the proposed operations.
- By using eclipse modelling framework (EMF) (Budinsky et al., 2004) to develop meta-model, we can explicitly define the workflow model. This helps us to give a clear visibility of the model. The code generation for EMF models can be adapted and in its default setting. It provides model change notification features in the event of a model change. EMF produces interfaces and a factory to generate objects and allows us to keep our model independent of implementation.

This EMF implementation also allows:

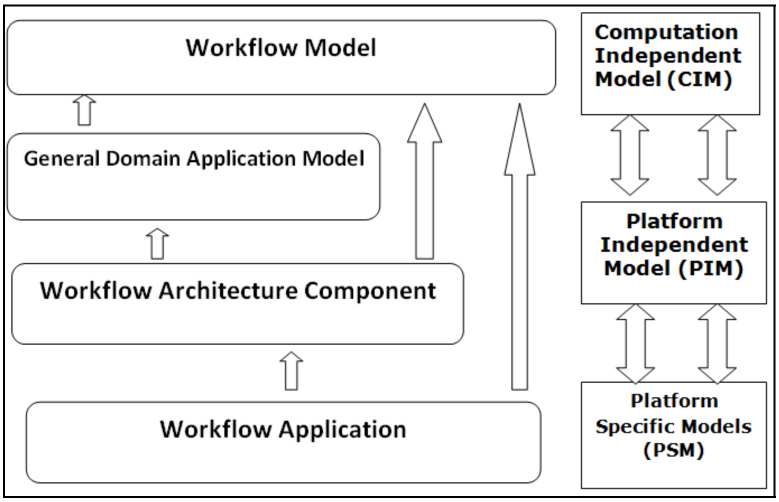
- Perform the endogenous transformation to make workflow model evolution, this endogenous transformation model affects models expressed in the same language (because ATL refine mode is more suitable with EMF technology).
- It is possible to create meta-model according to several views: in the form of ECORE diagrams.
- A corresponding XML schema and the associated serialisation/deserialisation operations.
- EMF fully accepts the OCL constraint language, which allows to considerably enrich ECORE meta-model and to perform constraint tests.
- It is easy to create a dynamic instance from a meta-model and make validations or modifications on it.

4 Our proposed approach

This section introduces our proposed approach and its concepts of MDA (see Figure 3). From a model of independent calculation (CIM) often abstract, such that, a workflow process or a functional description, the PIM is derived from the elaborations and mapping between the original concepts and rendered PIM. Once the PIM is refined and stable enough, the specific models to the platform (PSM) are derived through further development and refinement. The PSMs are transformed into operational systems with source code.

The CIM layer defines the generic concept of the domain, the constraints on the solution and the specific requirements.

Figure 3 Our proposed approach for workflow evolution


Workflow Model
General Domain Application Model
Workflow Architecture Component
Workflow Application
Computation Independent Model (CIM)
Platform Independent Model (PIM)
Platform Specific Models (PSM)

The artefacts of the CIM layer focus in large part on the system requirements and their environment to provide a vocabulary and an appropriate context (domain models, conceptual classes). The CIM lawyer does not contain any details of treatment or implementation.

Instead, it transmits non-functional requirements such as the constraints of business, of the constraints of deployment and the constraints of performance as well as functional constraints.

The PIM provides the architecture, the plan of logical design, but not the execution plan. Beyond the high level of service, the domain of the problem itself must be modelled from a perspective of treatment. The PIM is where the logical components of the system, their behaviours and interactions are modelled. The PIM artefacts focus on the modelling of what the system should do to an external perspective or logical.

The PIM is used to represent the business functionality of the system without the inclusion of technical aspects.

The mapping of the PIM to the PSM is an essential element of the MDA approach. Mappings of PIM representations to those which implement the features or functions directly in the technologies specific to the platform are the most important points in MDA. This mapping allows an orderly transition from one platform to another. But, the utility does not stop there. As the PIM, it is possible to have layers within the PSM to generate transformations intermediaries on the path of the executable system. These models range from models of behaviour detailed in source code used in the construction of the system.

We explored the ways to use MDA to facilitate the development of workflow, abstraction and programming at a higher level. The software production model approach suggests that the workflow change will be addressed at the appropriate abstraction level. In other words, if a change is made at the application level, it must be supported by the transformation and mapping process.

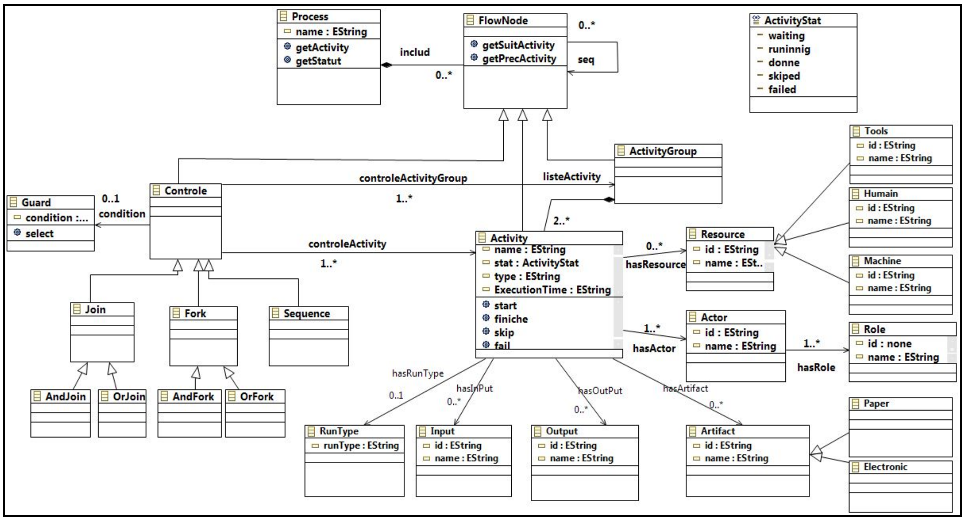
The meta-model of this generic workflow supports the description of many principal architectural elements that can capture all the features and functionality of the various workflows in the trade business domain and the different relations among these elements.

This meta-model supports the specification of relevant workflow aspects in the e-commerce domain, such as the decomposition of activities into sub-activities, the control flow and the data flow, as well as the assignment of activities to processing entities and the actors of each activity.

The class activity is the focal part. A process contains a many FlowNode that can be activity group, simple activity or control node. The class FlowNode has a reflexive association called *seq* to characterise the sequence relationship. An activity has other relations with *Resource* class, Actor class, *RunType* class, *Input* class, *Output* class and *Artifact* class.

The *Artifact* class has tow subclass *Paper* and *Electronic*, also the *Resource* class has three subclass: *Tools*, *Human* and *Machine*. The activity class has an attribute state associated with the enumeration type *State*, which identifies the execution state in which the activity is currently executed, this enumeration has five states (waiting, running, done, skipped and failed). To guarantee the control of various activities, we have the class *Control* that contains three sorts of control *Join*, *Fork* and *Sequence*. The condition for choosing the correct way is modelled by class *Guard* which is associated with the subsequent activities. At run-time, the user can choose the *Guard* with the *select*() operation. The meta-model is defined with EMF, the proposed meta-model is shown in Figure 4.

Figure 4 Meta-model of workflow in e-commerce business domain



In our approach, we have adopted two modes of transformation, horizontal transformation and vertical transformation.

- Horizontal transformation of model: Where the source and target models are both at the same level of abstraction (Agostinho et al., 2014; Kriouile et al., 2015). Examples of horizontal transformation:
  - a refactoring
  - b merging.
- Vertical transformation of model: In this transformation, different levels of abstraction are used in the source and target models (Agostinho et al., 2014; Kriouile et al., 2015; Mouheb et al., 2015). Examples of vertical transformation:
  - a refinement (specialisation)
  - b PIM to PSM transformations
  - c abstraction (generalisation).

To ensure this evolution and model transformation, we have used ATL rules to transform the model and to adapt these new concepts.

We have used the endogenous transformation model (Selim et al., 2017; Vara et al., 2010) that affects models expressed in the same language. The two source and target models are consistent with the same meta-model. We also used the refinement mode in ATL, in which this transformation mode works as if the implicit copy rules were mapped to explicit rules for each element. This copy rule generates a target element of the same kind as the source element. Next, it initiates all the properties of the new element by copying the values of the properties of the source element and generates a traceable relationship.

In our approach, we have used the refining mode in ATL and endogenous transformation procedures, to ensure progress and endogenous evolution in the workflow model.

We make principles of CIM level and change guidelines to PIM level. Every change is described in ATL rules before changing.

The workflow model must be adapted to its changing environment for example, new customer requirements or redesigned business processes. Therefore, the evolution of the workflow model, that is, the modification of the workflow model over time, must be supported. The evolution of the workflow model may include the creation of a new workflow concept as well as modification and deletion.

Evolutionary change is started by the company's management to improve efficiency or reactivity or is imposed by the legislator or the evolving requirements of the market. After that, the designers can refine the workflow model according to the evolution scenario proposed by company manager.

4.1 Refinement transformations

These transformations are used to refine a model towards an implementation (Perillo and Di Natale, 2017; Kouamou and Kungne, 2017), for example PIM to PSM transformations in the MDA. They generally eliminate some constructions or structures, for example multiple inheritances, from a model and represent them in place with constructions that are the same as those available in the implementation platform (Zhou et al., 2017):

- PIM to PSM

- PIM to PIM.

The designer can mediate only to choose the non-transformable components. At any case, in the beginning of every transformation, we verify through ATL if the component is transformable, for that in every manage, we call partners effectively portrayed with OCL.

We can make refinements in workflow models to obtain transformable models. Based on the workflow meta-model, we propose a set of dynamic change operations and designer intervention operations to support the controlled execution of complex activities.

4.2 Dynamic change operations

To perform a dynamic change operation, the designer calls a customised modelling activity. During this activity, all operations such as add activity, delete activity, add resource, delete resource and anthers operations will be executed and the original workflow model will be substituted by new workflow model. These operations are not only available during the initial creation of activity models, but also during the execution of workflow instances.

4.3 Endogenous rules in ATL which allow the transformation and evolution of workflow model

In this work, we have developed a library of endogenous rules in ATL allowing the transformation and evolution of the workflow model among these rules, we have:

- rules for the change and the modification of the descriptions of the classes
- rules for the insertion of the classes in the model
- the rules for the insertion of the relations between the relations of the classes
- the rules for the removal of the classes in the model
- the rules for the removal of the relations between the relations of the classes
- rule of the rules changes to the names of the classes in the model
- rules of deletion of the classes in the model
- many other rules.

These rules are proposed by the designer in a well-defined context. According to other processes, the designer must choose appropriate evolution rules.

Adaptations are applied based on changes made to the model, adaptations can be extended or even modified by designers, who can specify custom adaptation rules to refine or replace transformation adaptations. The refinement of models creates different adaptations, depending on the type of evolved model; each adaptation is saved in the library and can be customised to meet the different evolution scenarios.

Our approach is implemented at design-time and all evolutions should be validated by the designer. The designer can refine the workflow model according to the evolution scenario proposed by company manager. The designer always keeps control to validate or not the evolution produced. If the changes have not occurred, the system maintains the previous version of the model until the validation of the new version by the designer.

5 Implementation and case study

In this section, we present a case study on e-commerce domain to illustrate our approach and the transformation rules that enable the evolution of the workflow as well as the implementation of these rules, then its results after execution.

Before proceeding to the execution of the transformation rules, we have first created ECORE models (Sunyé, 2017) corresponding to our meta-model. In the second step, we have implemented the rules of ATL. The output of the XML Metadata Interchange (XMI) file, produced by this transformation, is transformed into an EMF model (Budinsky et al., 2004).

MDA standards and frameworks that define how to represent computers are XMI, Java Metadata Interface (JMI) and EMF. The XMI standard defines how to represent templates as an XML document. The JMI and the EMF define how to represent models as Java objects (Budinsky et al., 2004).

Thanks to the XMI and JMI standards and the EMF framework, it is possible to represent any model. XMI concretises the durability of the models in that it offers an XML representation format. JMI and EMF are the operational bases of MDA since they allow the construction of operations of productivity on the models. Let us specify that there are bridges between these two standards and that it is possible to pass from a representation to another without difficulty.

We have used EMF to store the model content via the EMF persistence framework. EMF uses XMI. XMI is a standard for sharing metadata information via XML, if we persist an EMF object, all dependent objects will automatically be persistent. Objects which do not have a 'contains relationship' must be added explicitly (Budinsky et al., 2004).

5.1 Example of process modelling

In this section, the meta-model in the previous section is used to create workflow model. In the first paragraph of the workflow is described in natural language, also in workflow model resources and data are not included in the model because they are not part of the meta-model. Only activities and flow control aspects are modelled.

5.1.1 Description of workflow example

The following example describes the workflow to make an order. We identify three actors in this system: the customer, the accounting department and delivery service, the actions are as follows.

The customer begins by making an order, giving rise to the drawing up of a quote by the quotation activity of the accounting department, which is itself broken down into two sub-activities. Check the availability of the products ordered and calculates the price of the order.

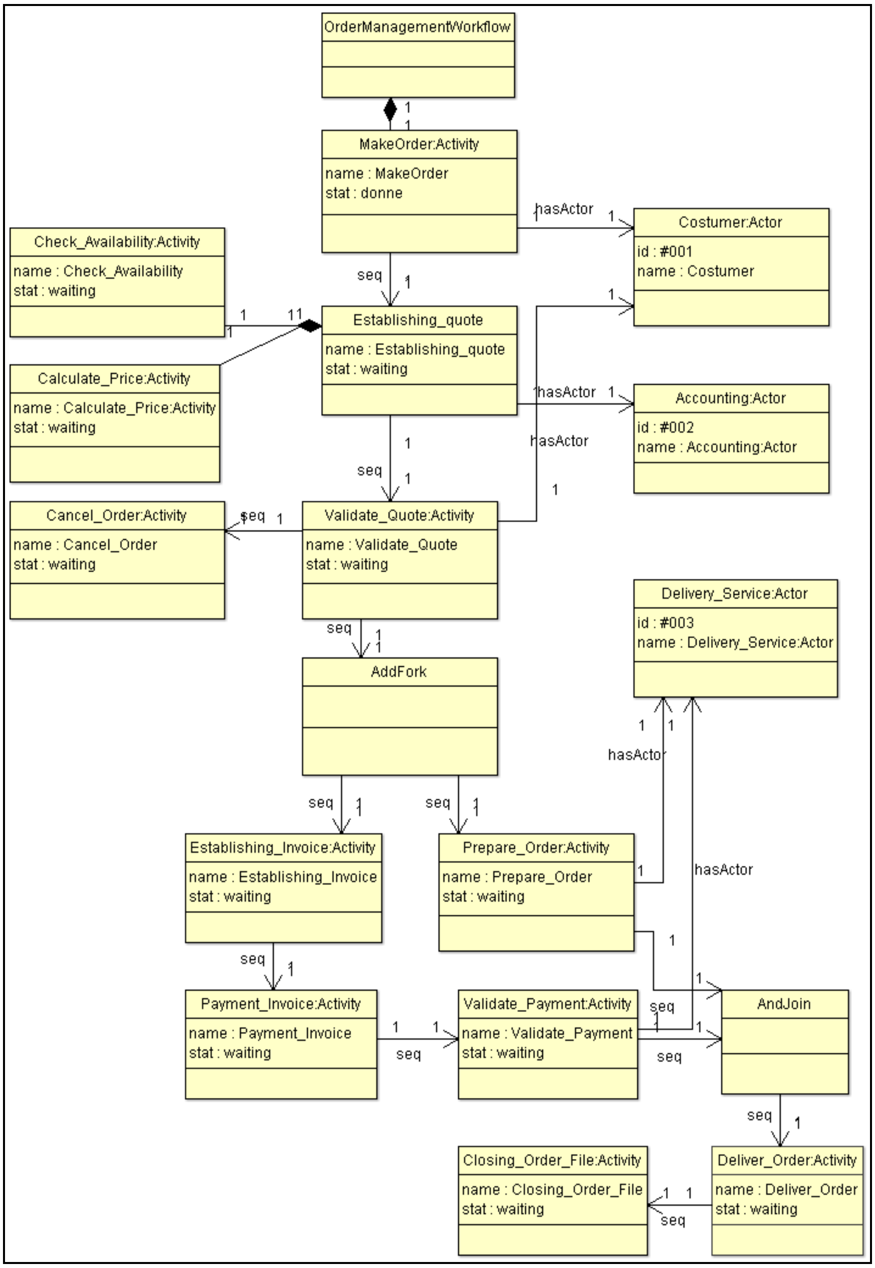
Quote is the name of a system class, data streams are typed the client can then change its order (return to initial activity to order), cancel (go to final state) or validate the quote.

If the customer validates the order, two actions can be taken:

- preparation of the order by the delivery service
- the processing of invoices and payment.

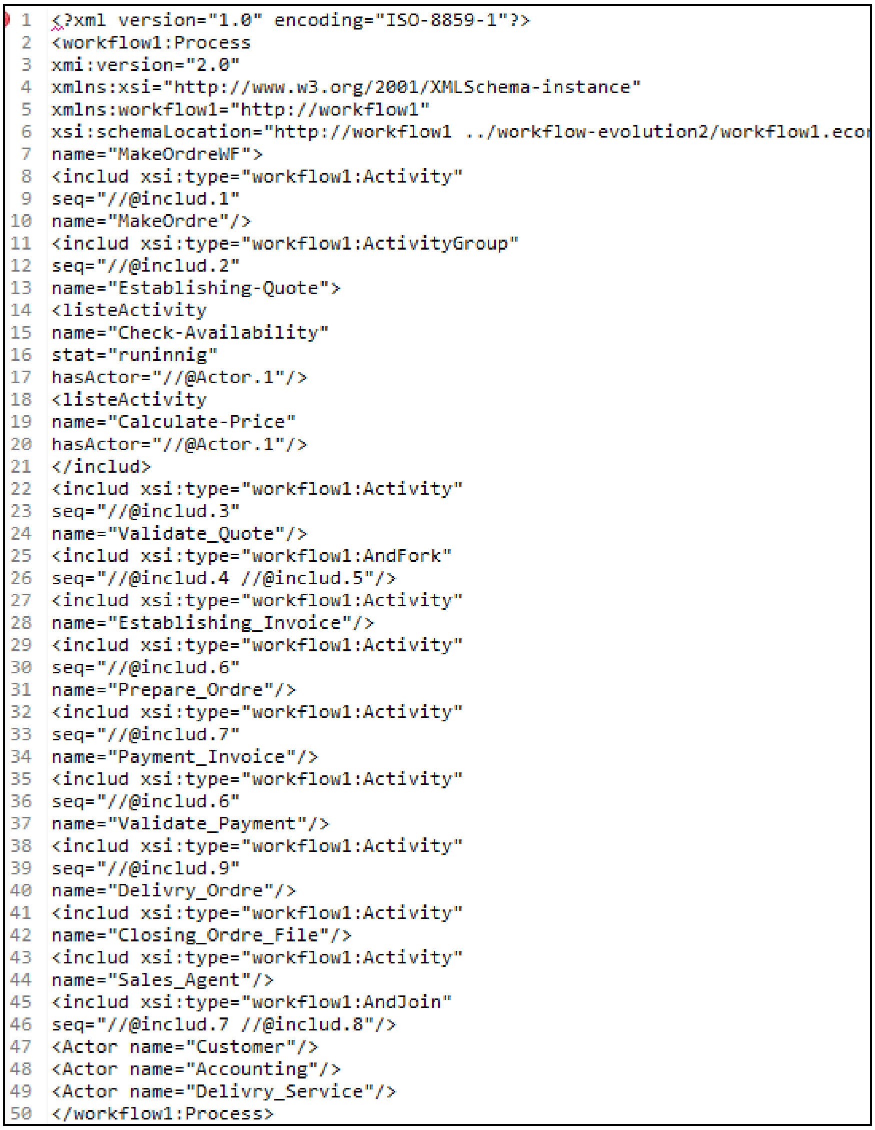
The accounting department creates invoices and sends it to the customer (invoices for objects sent to the sent state) which makes the payment and sends the invoice (which is present to the regulated state).

Figure 5 Workflow model before evolution (see online version for colours)



When the order is ready and confirmed the customer's payment (by synchronising the appointment of these two activities), the order is delivered to the customer and processing is completed order to define a workflow of treatment of order and it is applied to him of the different rules for a general evolution of its activities, Figure 5 shows the diagram of this workflow before evolution.

Figure 6 XMI file of workflow model before evolution



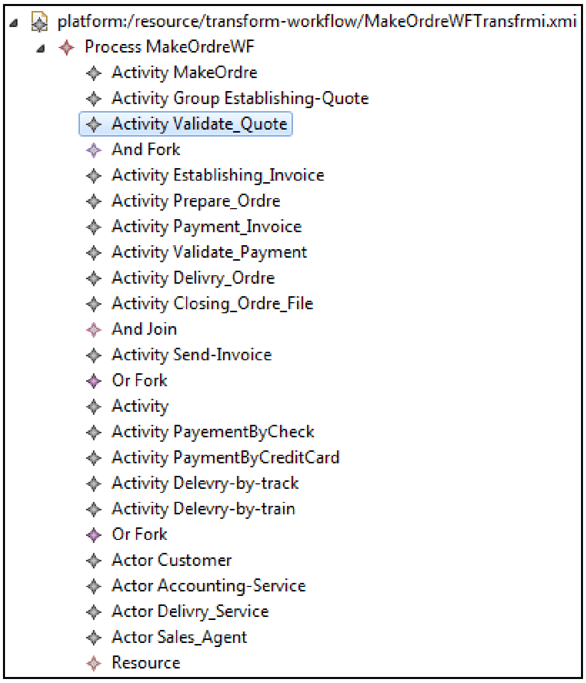
```xml
<?xml version="1.0" encoding="ISO-8859-1"?>
<workflow1:Process
xmi:version="2.0"
xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xmlns:workflow1="http://workflow1"
xsi:schemaLocation="http://workflow1 ../workflow-evolution2/workflow1.eco
name="MakeOrdreWF">
<includ xsi:type="workflow1:Activity"
seq="//@includ.1"
name="MakeOrdre"/>
<includ xsi:type="workflow1:ActivityGroup"
seq="//@includ.2"
name="Establishing-Quote">
<listeActivity
name="Check-Availability"
stat="runinnig"
hasActor="//@Actor.1"/>
<listeActivity
name="Calculate-Price"
hasActor="//@Actor.1"/>
</includ>
<includ xsi:type="workflow1:Activity"
seq="//@includ.3"
name="Validate_Quote"/>
<includ xsi:type="workflow1:AndFork"
seq="//@includ.4 //@includ.5"/>
<includ xsi:type="workflow1:Activity"
name="Establishing_Invoice"/>
<includ xsi:type="workflow1:Activity"
seq="//@includ.6"
name="Prepare_Ordre"/>
<includ xsi:type="workflow1:Activity"
seq="//@includ.7"
name="Payment_Invoice"/>
<includ xsi:type="workflow1:Activity"
seq="//@includ.6"
name="Validate_Payment"/>
<includ xsi:type="workflow1:Activity"
seq="//@includ.9"
name="Delivry_Ordre"/>
<includ xsi:type="workflow1:Activity"
name="Closing_Ordre_File"/>
<includ xsi:type="workflow1:Activity"
name="Sales_Agent"/>
<includ xsi:type="workflow1:AndJoin"
seq="//@includ.7 //@includ.8"/>
<Actor name="Customer"/>
<Actor name="Accounting"/>
<Actor name="Delivry_Service"/>
</workflow1:Process>
```

Before preceding the execution of transformation rules, we first created ECORE models corresponding to our meta-model. In the second step, we have implemented the rules of ATL. The result of XMI file, generated by this transformation, is converted to an EMF model. The meta-model is defined with EMF (Budinsky et al., 2004).

5.2 Example of workflow evolution scenario

In this example, we present a scenario of evolution of workflow Make Order, these changes are proposed by the company's managers to improve the workflow model.

Figure 7 Workflow to make an order before evolution (see online version for colours)



```
platform:/resource/transform-workflow/MakeOrdreWFTransfrmi.xmi
  Process MakeOrdreWF
    Activity MakeOrdre
    Activity Group Establishing-Quote
    Activity Validate_Quote
    And Fork
    Activity Establishing_Invoice
    Activity Prepare_Ordre
    Activity Payment_Invoice
    Activity Validate_Payment
    Activity Delivry_Ordre
    Activity Closing_Ordre_File
    And Join
    Activity Send-Invoice
    Or Fork
    Activity
    Activity PayementByCheck
    Activity PaymentByCreditCard
    Activity Delevry-by-track
    Activity Delevry-by-train
    Or Fork
    Actor Customer
    Actor Accounting-Service
    Actor Delivry_Service
    Actor Sales_Agent
    Resource
```

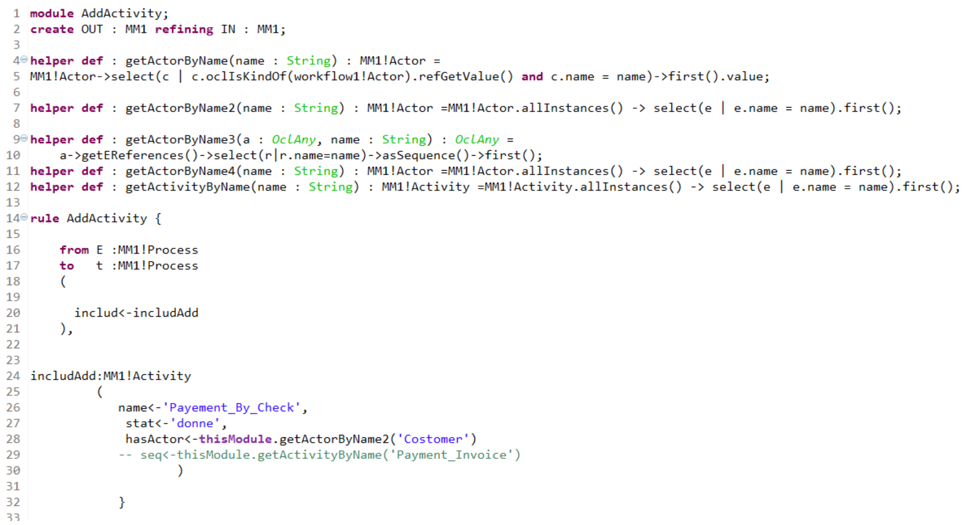
Assume that after several instances of the workflow version, the company's management decides to improve it in the following ways:

- Add the actor 'Sale-Agent': In the new workflow version, the company wants to add a sales agent to improve the relationship with the customer. This agent has several activities for example:
  a establish customer files
  b respond to customer needs
  c communicate information about products or services to customers
  d make the sale from beginning to end.
- Add the actor 'Sales-Agent' to the activity 'Prepare-Order': In order to associate the actor 'Sale-agent' with the activity 'Prepare-Order'.
- Delete activity 'Check-Availability' and 'Calculate Price': To merge these two activities into the 'Establishing_quote' activity in order to accelerate the processing of the customer request.
- Add Role 'Control' to actor 'Sale-Agent'.

- Add activity ‘Send-Invoice’ before the activity ‘Establishing-Invoice’: To inform the customer more quickly and be more close to the customer.
- Add the Or-Fork control node between ‘Payment-by-credit’ and ‘By-Check’: To add an Or-Fork control node to make the choice in the payment method either by ‘Payment-by-credit’ or by ‘By-Check’.
- Add the Or-Fork control node between ‘Delivery-by-track’ and ‘by-train’: To add an OR-fork control node to make the choice in the delivery method either by ‘Delivery-by-track’ or by ‘by-train’.
- Change of the name ‘Accounting’ to ‘Accounting-Service’: To give a new name to the actor ‘Accounting’.

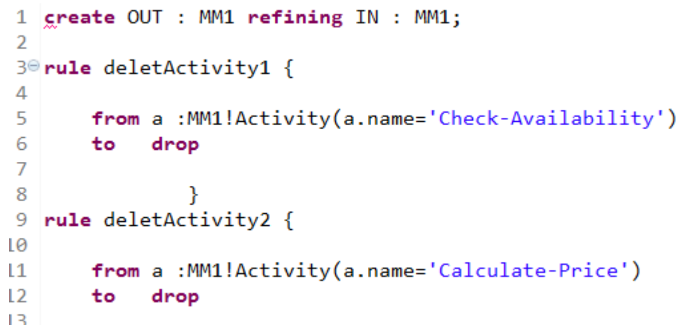
In our example, we used this scenario to test the evolution approach and the ATL rules. In the following section, we present the different ATL rules of the endogenous transformation to perform this evolution.

Listing 1 Rule to add new activity ‘Payement_By_Check’ (see online version for colours)



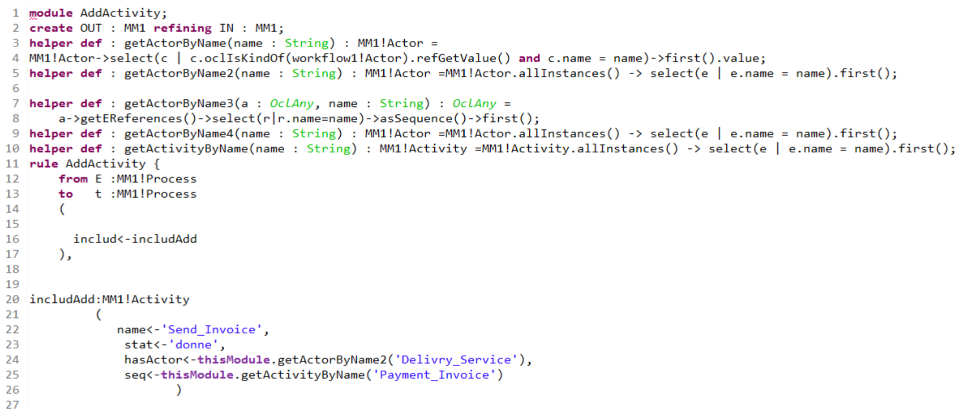
```
module AddActivity;
create OUT : MM1 refining IN : MM1;

helper def : getActorByName(name : String) : MM1!Actor =
MM1!Actor->select(c | c.oclIsKindOf(workflow1!Actor).refGetValue() and c.name = name)->first().value;

helper def : getActorByName2(name : String) : MM1!Actor =MM1!Actor.allInstances() -> select(e | e.name = name).first();

helper def : getActorByName3(a : OclAny, name : String) : OclAny =
    a->getEReferences()->select(r|r.name=name)->asSequence()->first();
helper def : getActorByName4(name : String) : MM1!Actor =MM1!Actor.allInstances() -> select(e | e.name = name).first();
helper def : getActivityByName(name : String) : MM1!Activity =MM1!Activity.allInstances() -> select(e | e.name = name).first();

rule AddActivity {

    from E :MM1!Process
    to   t :MM1!Process
    (

      includ<-includAdd
    ),


includAdd:MM1!Activity
          (
             name<-'Payement_By_Check',
              stat<-'donne',
              hasActor<-thisModule.getActorByName2('Costomer')
             -- seq<-thisModule.getActivityByName('Payment_Invoice')
                    )

             }

```

Listing 2 Rule to delete activities ‘Check-Availability’ and ‘Calculate-Price’ (see online version for colours)



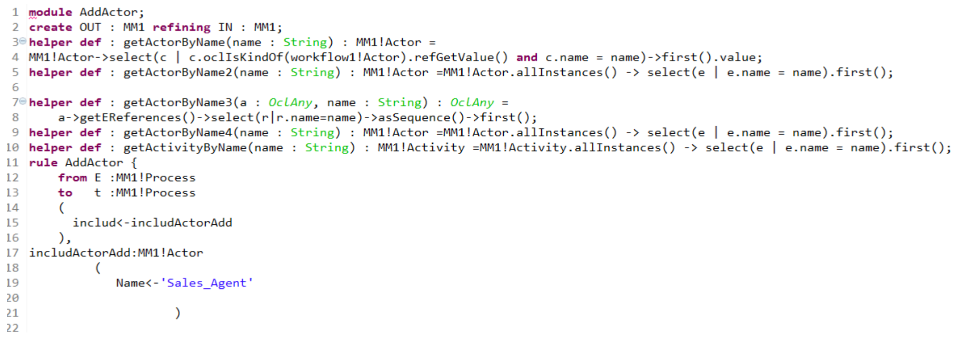
```
create OUT : MM1 refining IN : MM1;

rule deletActivity1 {

    from a :MM1!Activity(a.name='Check-Availability')
    to   drop

                 }
rule deletActivity2 {

    from a :MM1!Activity(a.name='Calculate-Price')
    to   drop

```

Listing 3 Rule to add a new activity ‘Send_Invoice’ (see online version for colours)



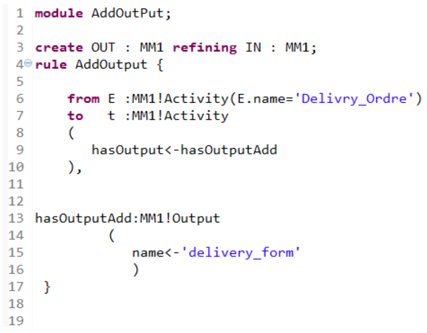
```
module AddActivity;
create OUT : MM1 refining IN : MM1;
helper def : getActorByName(name : String) : MM1!Actor =
MM1!Actor->select(c | c.oclIsKindOf(workflow1!Actor).refGetValue() and c.name = name)->first().value;
helper def : getActorByName2(name : String) : MM1!Actor =MM1!Actor.allInstances() -> select(e | e.name = name).first();

helper def : getActorByName3(a : OclAny, name : String) : OclAny =
    a->getEReferences()->select(r|r.name=name)->asSequence()->first();
helper def : getActorByName4(name : String) : MM1!Actor =MM1!Actor.allInstances() -> select(e | e.name = name).first();
helper def : getActivityByName(name : String) : MM1!Activity =MM1!Activity.allInstances() -> select(e | e.name = name).first();
rule AddActivity {
    from E :MM1!Process
    to   t :MM1!Process
    (

      includ<-includAdd
    ),


includAdd:MM1!Activity
         (
            name<-'Send_Invoice',
             stat<-'donne',
             hasActor<-thisModule.getActorByName2('Delivry_Service'),
             seq<-thisModule.getActivityByName('Payment_Invoice')
                  )

```

Listing 4 Rule to add a new actor ‘Sales-agent’ (see online version for colours)



```
module AddActor;
create OUT : MM1 refining IN : MM1;
helper def : getActorByName(name : String) : MM1!Actor =
MM1!Actor->select(c | c.oclIsKindOf(workflow1!Actor).refGetValue() and c.name = name)->first().value;
helper def : getActorByName2(name : String) : MM1!Actor =MM1!Actor.allInstances() -> select(e | e.name = name).first();

helper def : getActorByName3(a : OclAny, name : String) : OclAny =
    a->getEReferences()->select(r|r.name=name)->asSequence()->first();
helper def : getActorByName4(name : String) : MM1!Actor =MM1!Actor.allInstances() -> select(e | e.name = name).first();
helper def : getActivityByName(name : String) : MM1!Activity =MM1!Activity.allInstances() -> select(e | e.name = name).first();
rule AddActor {
    from E :MM1!Process
    to   t :MM1!Process
    (
      includ<-includActorAdd
    ),
includActorAdd:MM1!Actor
         (
            Name<-'Sales_Agent'

                    )

```

Listing 5 Rule to add a new output ‘Delivry_Ordre’ (see online version for colours)



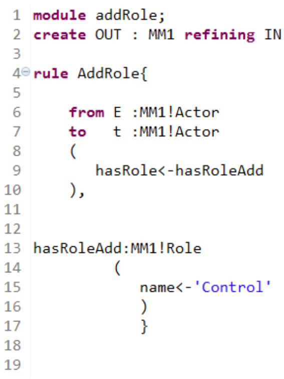
```
module AddOutPut;

create OUT : MM1 refining IN : MM1;
rule AddOutput {

    from E :MM1!Activity(E.name='Delivry_Ordre')
    to   t :MM1!Activity
    (
       hasOutput<-hasOutputAdd
    ),


hasOutputAdd:MM1!Output
         (
            name<-'delivery_form'
            )
 }


```

To execute this transformation, we have developed a software application with a rich user interface. This application allows:

- The selection of source model.
- The selection of target model.
- The selection of the ATL rules (from the library of ATL according to the evolution scenario).

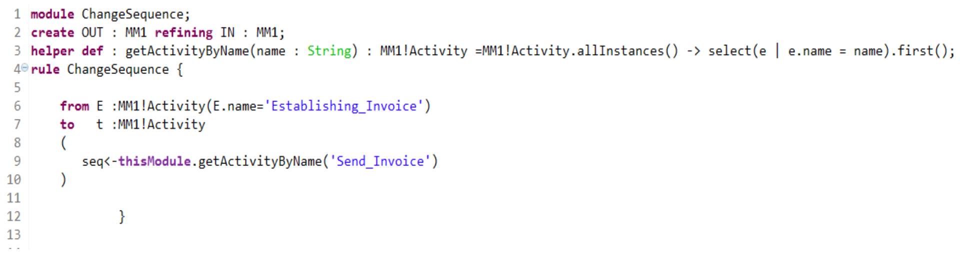
- The updating of the ATL rules in case of modification.
- The execution of transformation.
- The selection of input model before the evolution.
- The selection of output model after evolution.
- The display XMI format of input model before the evolution.
- The display XMI format of output model after the evolution.
- The display EMF format of input model before the evolution.
- The display EMF format of the output model after the evolution.

Listing 6 Rule to add a role ‘Control’ (see online version for colours)



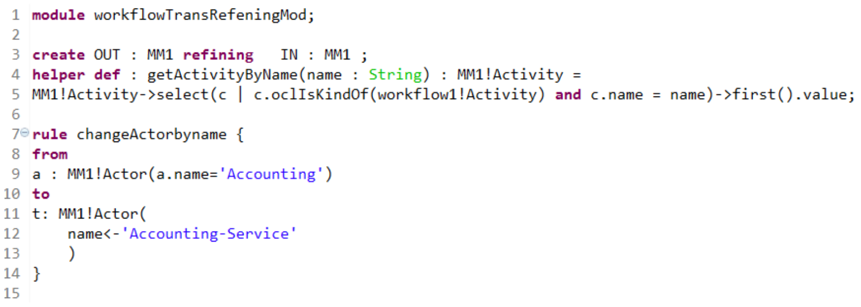
```
module addRole;
create OUT : MM1 refining IN : MM1;

rule AddRole{

    from E :MM1!Actor
    to   t :MM1!Actor
    (
       hasRole<-hasRoleAdd
    ),


hasRoleAdd:MM1!Role
         (
            name<-'Control'
            )
            }


```

Listing 7 Rule to change the sequence (see online version for colours)



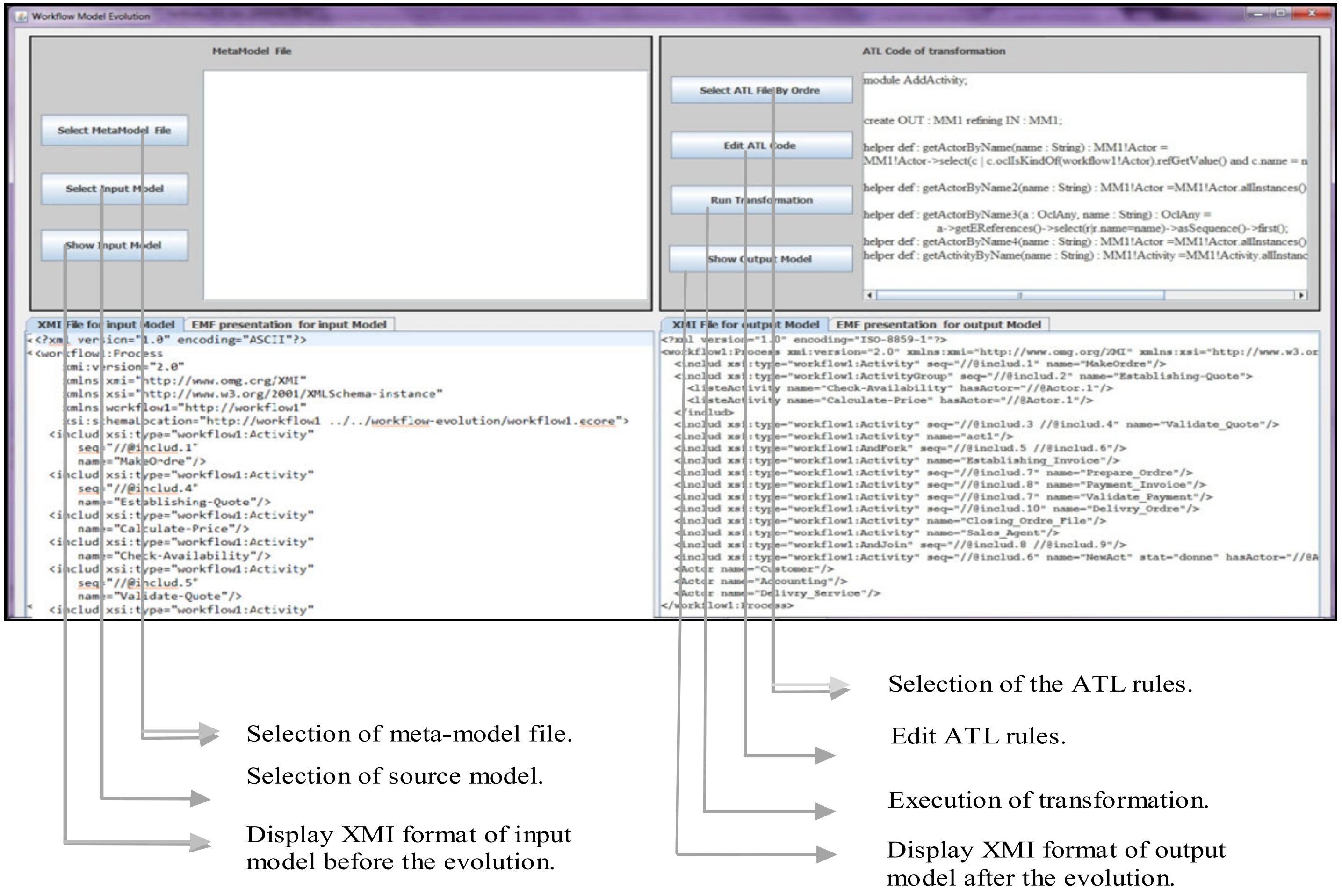
```
module ChangeSequence;
create OUT : MM1 refining IN : MM1;
helper def : getActivityByName(name : String) : MM1!Activity =MM1!Activity.allInstances() -> select(e | e.name = name).first();
rule ChangeSequence {

   from E :MM1!Activity(E.name='Establishing_Invoice')
   to   t :MM1!Activity
   (
      seq<-thisModule.getActivityByName('Send_Invoice')
   )

          }

```

Listing 8 Rule to change the actor’s name from ‘Accounting’ to ‘Accounting-Service’ (see online version for colours)



```
module workflowTransRefeningMod;

create OUT : MM1 refining   IN : MM1 ;
helper def : getActivityByName(name : String) : MM1!Activity =
MM1!Activity->select(c | c.oclIsKindOf(workflow1!Activity) and c.name = name)->first().value;

rule changeActorbyname {
from
a : MM1!Actor(a.name='Accounting')
to
t: MM1!Actor(
    name<-'Accounting-Service'
    )
}

```

Figure 8 Software to execute workflow evolution (see online version for colours)

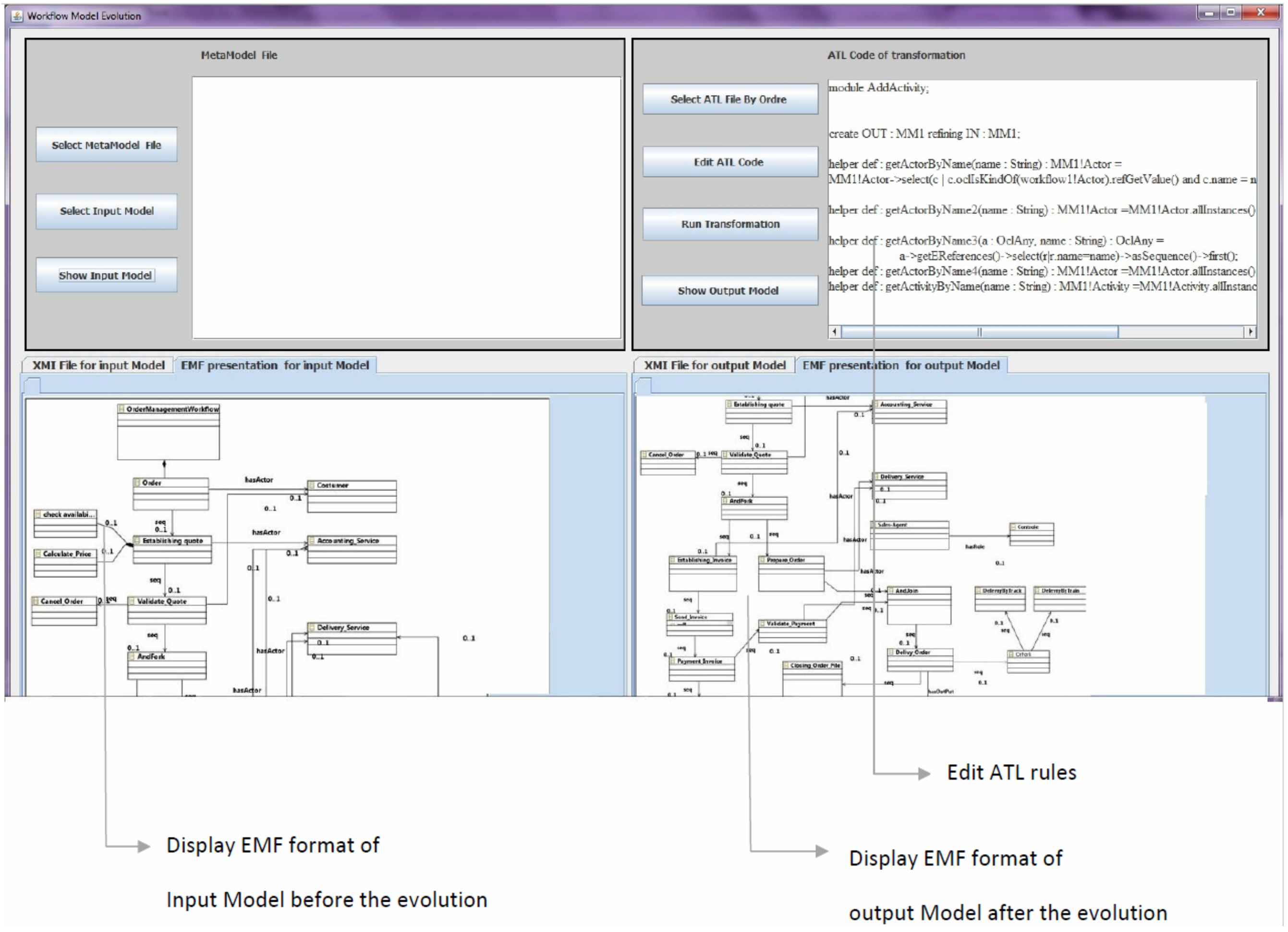

Workflow Model Evolution
MetaModel File
Select MetaModel File
Select Input Model
Show Input Model
ATL Code of transformation
Select ATL File By Ordre
Edit ATL Code
Run Transformation
Show Output Model
XMI File for input Model
EMF presentation for input Model
XMI File for output Model
EMF presentation for output Model
Selection of meta-model file.
Selection of source model.
Display XMI format of input model before the evolution.
Selection of the ATL rules.
Edit ATL rules.
Execution of transformation.
Display XMI format of output model after the evolution.

```
module AddActivity;


create OUT : MM1 refining IN : MM1;

helper def : getActorByName(name : String) : MM1!Actor =
MM1!Actor->select(c | c.oclIsKindOf(workflow1!Actor).refGetValue() and c.name = n

helper def : getActorByName2(name : String) : MM1!Actor =MM1!Actor.allInstances()

helper def : getActorByName3(a : OclAny, name : String) : OclAny =
                a->getEReferences()->select(r|r.name=name)->asSequence()->first();
helper def : getActorByName4(name : String) : MM1!Actor =MM1!Actor.allInstances()
helper def : getActivityByName(name : String) : MM1!Activity =MM1!Activity.allInstanc
```

```
<?xml version="1.0" encoding="ASCII"?>
<workflow1:Process
    xmi:version="2.0"
    xmlns:xmi="http://www.omg.org/XMI"
    xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
    xmlns:workflow1="http://workflow1"
    xsi:schemaLocation="http://workflow1 ../../workflow-evolution/workflow1.ecore">
  <includ xsi:type="workflow1:Activity"
      seq="//@includ.1"
      name="MakeOrdre"/>
  <includ xsi:type="workflow1:Activity"
      seq="//@includ.4"
      name="Establishing-Quote"/>
  <includ xsi:type="workflow1:Activity"
      name="Calculate-Price"/>
  <includ xsi:type="workflow1:Activity"
      name="Check-Availability"/>
  <includ xsi:type="workflow1:Activity"
      seq="//@includ.5"
      name="Validate-Quote"/>
  <includ xsi:type="workflow1:Activity"
```

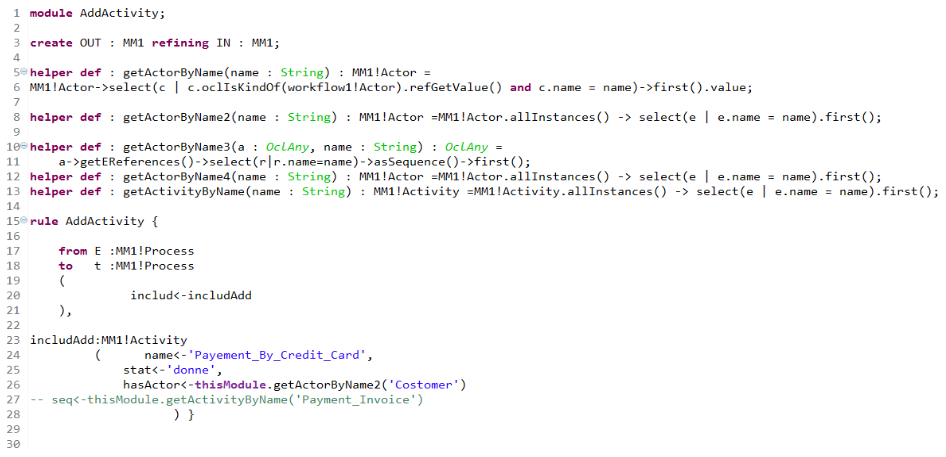
```
<?xml version="1.0" encoding="ISO-8859-1"?>
<workflow1:Process xmi:version="2.0" xmlns:xmi="http://www.omg.org/XMI" xmlns:xsi="http://www.w3.or
  <includ xsi:type="workflow1:Activity" seq="//@includ.1" name="MakeOrdre"/>
  <includ xsi:type="workflow1:ActivityGroup" seq="//@includ.2" name="Establishing-Quote">
    <listeActivity name="Check-Availability" hasActor="//@Actor.1"/>
    <listeActivity name="Calculate-Price" hasActor="//@Actor.1"/>
  </includ>
  <includ xsi:type="workflow1:Activity" seq="//@includ.3 //@includ.4" name="Validate_Quote"/>
  <includ xsi:type="workflow1:Activity" name="act1"/>
  <includ xsi:type="workflow1:AndFork" seq="//@includ.5 //@includ.6"/>
  <includ xsi:type="workflow1:Activity" name="Establishing_Invoice"/>
  <includ xsi:type="workflow1:Activity" seq="//@includ.7" name="Prepare_Ordre"/>
  <includ xsi:type="workflow1:Activity" seq="//@includ.8" name="Payment_Invoice"/>
  <includ xsi:type="workflow1:Activity" seq="//@includ.7" name="Validate_Payment"/>
  <includ xsi:type="workflow1:Activity" seq="//@includ.10" name="Delivry_Ordre"/>
  <includ xsi:type="workflow1:Activity" name="Closing_Ordre_File"/>
  <includ xsi:type="workflow1:Activity" name="Sales_Agent"/>
  <includ xsi:type="workflow1:AndJoin" seq="//@includ.8 //@includ.9"/>
  <includ xsi:type="workflow1:Activity" seq="//@includ.6" name="NewAct" stat="donne" hasActor="//@A
  <Actor name="Customer"/>
  <Actor name="Accounting"/>
  <Actor name="Delivry_Service"/>
</workflow1:Process>
```

Figure 9 Software to execute workflow evolution (see online version for colours)



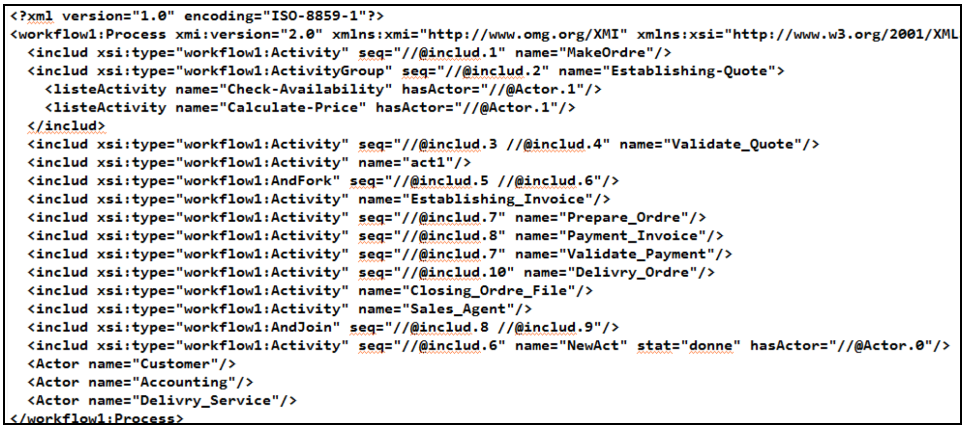
```
module AddActivity;


create OUT : MM1 refining IN : MM1;

helper def : getActorByName(name : String) : MM1!Actor =
MM1!Actor->select(c | c.oclIsKindOf(workflow1!Actor).refGetValue() and c.name = n

helper def : getActorByName2(name : String) : MM1!Actor =MM1!Actor.allInstances()

helper def : getActorByName3(a : OclAny, name : String) : OclAny =
            a->getEReferences()->select(r|r.name=name)->asSequence()->first();
helper def : getActorByName4(name : String) : MM1!Actor =MM1!Actor.allInstances()
helper def : getActivityByName(name : String) : MM1!Activity =MM1!Activity.allInstanc
```

Listing 9 Rule to add the activity ‘Payement_By_Credit_Card’ (see online version for colours)



```
module AddActivity;

create OUT : MM1 refining IN : MM1;

helper def : getActorByName(name : String) : MM1!Actor =
MM1!Actor->select(c | c.oclIsKindOf(workflow1!Actor).refGetValue() and c.name = name)->first().value;

helper def : getActorByName2(name : String) : MM1!Actor =MM1!Actor.allInstances() -> select(e | e.name = name).first();

helper def : getActorByName3(a : OclAny, name : String) : OclAny =
    a->getEReferences()->select(r|r.name=name)->asSequence()->first();
helper def : getActorByName4(name : String) : MM1!Actor =MM1!Actor.allInstances() -> select(e | e.name = name).first();
helper def : getActivityByName(name : String) : MM1!Activity =MM1!Activity.allInstances() -> select(e | e.name = name).first();

rule AddActivity {

    from E :MM1!Process
    to   t :MM1!Process
    (
               includ<-includAdd
    ),

includAdd:MM1!Activity
          (      name<-'Payement_By_Credit_Card',
               stat<-'donne',
               hasActor<-thisModule.getActorByName2('Costomer')
-- seq<-thisModule.getActivityByName('Payment_Invoice')
                     ) }


```

Figure 10 XMI file of workflow model after evolution



```
<?xml version="1.0" encoding="ISO-8859-1"?>
<workflow1:Process xmi:version="2.0" xmlns:xmi="http://www.omg.org/XMI" xmlns:xsi="http://www.w3.org/2001/XML
  <includ xsi:type="workflow1:Activity" seq="//@includ.1" name="MakeOrdre"/>
  <includ xsi:type="workflow1:ActivityGroup" seq="//@includ.2" name="Establishing-Quote">
    <listeActivity name="Check-Availability" hasActor="//@Actor.1"/>
    <listeActivity name="Calculate-Price" hasActor="//@Actor.1"/>
  </includ>
  <includ xsi:type="workflow1:Activity" seq="//@includ.3 //@includ.4" name="Validate_Quote"/>
  <includ xsi:type="workflow1:Activity" name="act1"/>
  <includ xsi:type="workflow1:AndFork" seq="//@includ.5 //@includ.6"/>
  <includ xsi:type="workflow1:Activity" name="Establishing_Invoice"/>
  <includ xsi:type="workflow1:Activity" seq="//@includ.7" name="Prepare_Ordre"/>
  <includ xsi:type="workflow1:Activity" seq="//@includ.8" name="Payment_Invoice"/>
  <includ xsi:type="workflow1:Activity" seq="//@includ.7" name="Validate_Payment"/>
  <includ xsi:type="workflow1:Activity" seq="//@includ.10" name="Delivry_Ordre"/>
  <includ xsi:type="workflow1:Activity" name="Closing_Ordre_File"/>
  <includ xsi:type="workflow1:Activity" name="Sales_Agent"/>
  <includ xsi:type="workflow1:AndJoin" seq="//@includ.8 //@includ.9"/>
  <includ xsi:type="workflow1:Activity" seq="//@includ.6" name="NewAct" stat="donne" hasActor="//@Actor.0"/>
  <Actor name="Customer"/>
  <Actor name="Accounting"/>
  <Actor name="Delivry_Service"/>
</workflow1:Process>
```

To perform this evolution, we have used ATL refining mode transformation, this transformation follows the in-place transformations semantics (Boronat, 2018).

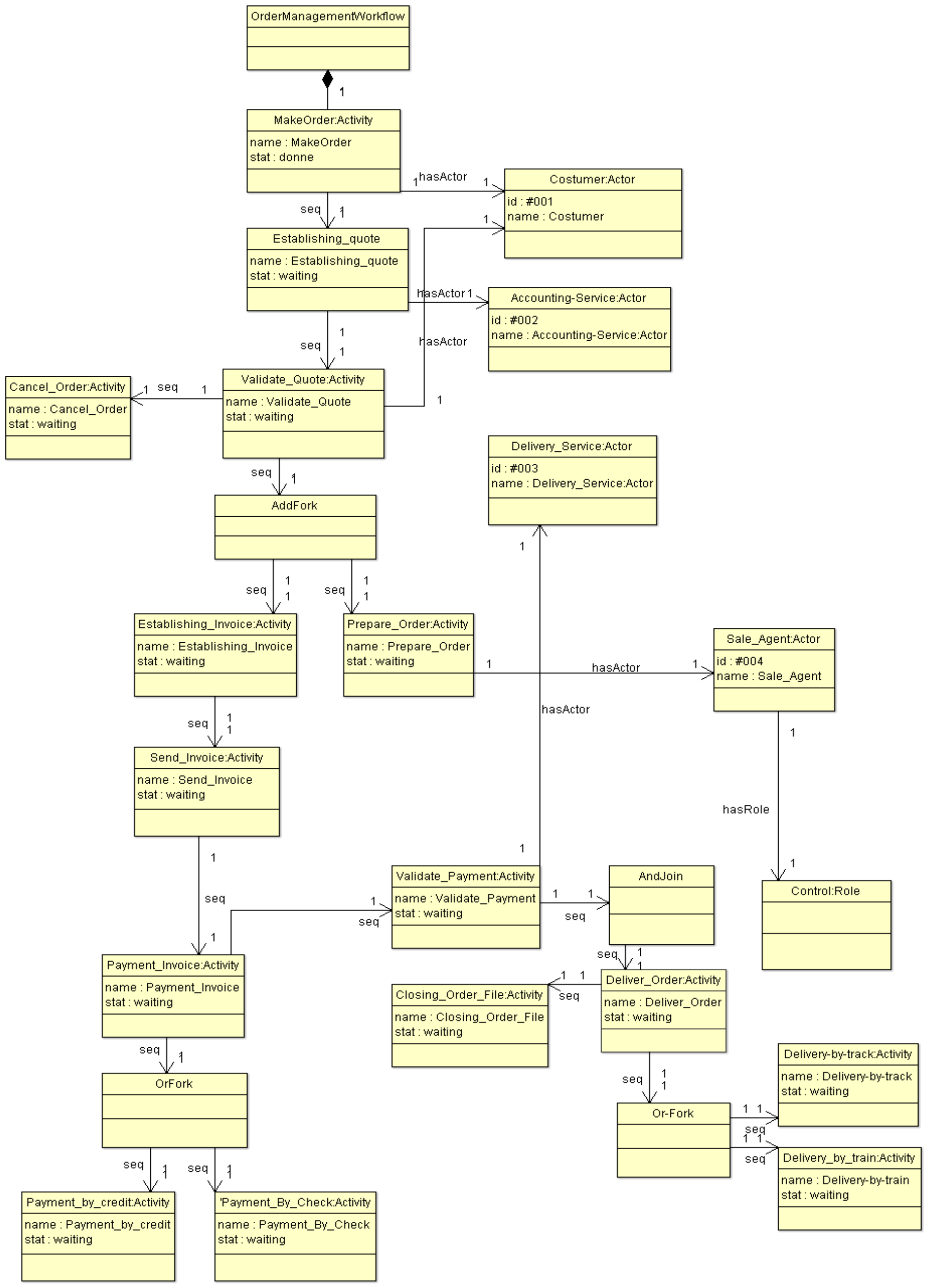
In this transformation mode, the source model is first copied to the target model and then transformation rules are applied. This mode of transformation can be thought as having a set of implicit rules copying all elements (meta-model dependant) and a set of explicit rules proposed by the designer (Boronat, 2018).

Executing an ATL refining mode transformation model needs several steps, it is first analysed and transformed to a model defined using the meta-model. This model is then controlled statically for semantic errors against the meta-models. The next step is the interpretation followed by an execution.

Our mechanism for executing ATL rules requires implicit triggering these rules in a specified order. The rules to be performed and their order of invocation must be selected according to the evolution scenario.

Moreover, ATL does not allow the automatic identification of rules at run-time as well as its call scheduling. For this reason, once the rules are written in ATL, we convert them to Java code in order to perform the evolution execution, from this generated code for each passing rule, we only need to manipulate the part of the Java code corresponding to the ATL rule in order to execute it.

Figure 11 Workflow model after evolution (see online version for colours)



6 Conclusions

Evolution of workflow models is a challenging task because it is regularly related to changes in trade agreements and business process organisation methods.

In this paper, we have proposed our new approach for workflow evolution using MDA technologies. In this approach, the workflow model is described by a specification at a very high level of abstraction using meta-model concept, to implement this specification a workflow meta-model has been introduced that allows to capture various aspects of workflows, this meta-model comprises concepts to capture functional, structural and informational, behavioural of workflows.

In this work, we have created a rule library to make the evolution. Evolution is applied based on changes made to the model and can be extended or even modified by designers, who can specify customised evolution rules to refine the model, to implement this library we have used endogenous rules in ATL, which allow the transformation and evolution of workflow model, this endogenous transformation model affects models expressed in the same language.

The two source and target models are consistent with the same meta-model. Among these rules, we have rules for changing and modifying model concepts, rules for inserting new concepts into the model and the relationships between its concepts, rules for deleting model concepts and many other rules.

We also show how workflow models can be automatically adapted through such model transformations. Based on MDA, our approach provides an efficient solution to workflow models transformation based on a workflow meta-model and proposes a set of element change operations and generic transformation that is not destined to a particular field.

Although, these achieved results, our work does not include certain points, which we want to integrate into future work among these points, the meta-model that we have developed is not rich enough to cover all the details in e-commerce domain. And we do not have enough rules to cover all evolution scenarios. Also, there is not a checking mechanism for proving model after evolution. Furthermore, the transformation rules are not easy to manipulate by the company's managers, they have a hard technical specification for them and must be manipulated by a modelling specialist.

As future work, we are planning to evaluate our proposed framework to inquire its performances and test it with some other domain and extend our meta-model in order to cover other areas of the business process. Another work we need to do is to enrich the library of ATL rules to cover more cases.

We will also study the possibilities to add a technique for proving model after evolution. The properties we are concerned about relate the structure of an input model with the structure of the transformed model. The main purpose of this feather is to prove the models after evolution. The designer may be certain about the structural soundness of the results of his transformations. In addition, we will try to simplify the manipulation of transformation rules in order to help non-expert users improve the quality of their workflow models and improve the user interface by adding more options and functionalities.

References

Adams, M., Wynn, M.T., Ouyang, C. and ter Hofstede, A.H. (2015) 'Realisation of cost-informed process support within the YAWL workflow environment', *in Asia-Pacific Conference on Business Process Management*, Springer, Cham, June, pp.3–18.

Agostinho, C., Bazoun, H., Zacharewicz, G., Ducq, Y. and Boye, H. (2014) 'Information models and transformation principles applied to servitization of manufacturing and service systems design', *in 2014 2nd International Conference on Model-driven Engineering and Software Development (MODELSWARD)*, IEEE, January, pp.657–665.

Allehyani, B. and Reiff-Marganiec, S. (2016) 'Towards ensuring a correct dynamic adaptation of workflows', *in Proceedings of the Eighth Saudi Students Conference in the UK*, pp.123–132.

Ben Fraj, I., Yousra, B.H. and BenAyed, L.J. (2017) 'A specification and execution approach of flexible cloud service workflow based on a meta model transformation', *in Proceedings of the 19th International Conference on Enterprise Information Systems (ICEIS 2017*), Vol. 2, pp.467–473.

Boronat, A. (2018) 'A formal framework for prototyping executable semantics in ATL', *in International Conference on Theory and Practice of Model Transformations*, Springer, Cham, June, pp.157–172.

Budinsky, F., Steinberg, D., Ellersick, R. and Grose, T. (2004) *Eclipse Modeling Framework*, Addison Wesley Professional, Boston, ISBN: 0131425420.

Casati F., Ceri S., Pernici B. and Pozzi G. (1996) 'Workflow evolution', *Proceedings of 15th International Conference on Conceptual Modeling (ER'96)*, Germany, October, pp.438–455.

Dohring, M. and Zimmermann, B. (2011) 'vBPMN: event-aware workflow variants by weaving BPMN2 and business rules', *in Proceedings of the 16th International Conference on Exploring Modeling Methods in Systems Analysis and Design, Ser. LNBIP*, Springer, Vol. 81, pp.332–341.

El Fazziki, A. et al. (2014) 'An agent based framework for an adaptive workflow management: model driven approach', *Journal of Convergence Information Technology (JCIT)*, September, Vol. 9, No. 5, pp.1–5.

Fraj, I.B., Hlaoui, Y.B. and Ayed, L.J.B. (2017) 'A modeling approach for flexible workflow applications of cloud services', *in 2017 IEEE 41st Annual Computer Software and Applications Conference (COMPSAC)*, IEEE, July, Vol. 2, pp.175–180.

Greiner, U., Mueller, R., Rahm, E., Ramsch, J., Heller, B. and Loeffler, M. (2005) 'AdaptFlow: protocol-based medical treatment using adaptive workflows', *Methods of Information in Medicine – Methodik der Information in der Medizin*, Vol. 44, No. 1, pp.80–88.

Hamri, S., Boudjlida, N. and Boufaida, M. (2007) 'An approach for building an OWL ontology for workflow interoperability', in *Enterprise Interoperability II*, pp.357–360, Springer, London.

Han, Y., Sheth, A. and Bussler, C. (1998) 'A taxonomy of adaptive workflow management', *in Workshop of the 1998 ACM Conference on Computer Supported Cooperative Work*, November, pp.1–11.

Hehenberger, P., Bricogne, M., Duigou, J.L., Zheng, C. and Eynard, B. (2016) 'Using meta-models to manage information change in the design process of systems of systems', *International Journal of Product Lifecycle Management*, Vol. 9, No. 4, pp.285–304.

IBM (1996) *IBM FlowMark: Modeling Workflow*, Version 2, Release 2, Publ. No. SH-19-8241-01.

Jouault, F. and Kurtev, I. (2006) 'Transforming models with ATL', *Model Transformations in Practice Workshop at MoDELS 2005*, Montego Bay, Jamaica.

Kalyanasundaram, P. and Ugale, S.P. (2015) 'Model transformation: concept, current trends and challenges', *International Journal of Computer Applications*, Vol. 119, No. 14, pp.1–5.

Kim, H.K., Yeo, H., Kim, T.H., Ramos, C., Marreiros, G. and Hwang, H.J. (2016) 'Developmental approaches covering context area mobile applications service oriented architecture and model driven architecture', *International Journal of Future Generation Communication and Networking*, Vol. 9, No. 12, pp.329–338.

Kiu, C-C. and Lee, C-S. (2017) 'E-commerce market trends: a case study in leveraging Web 2.0 technologies to gain and improve competitive advantage', *Int. J. Business Information Systems*, Vol. 25, No. 3, pp.373–392.

Kouamou, G.E. and Kungne, W.K. (2017) 'A structural and generative approach to multilayered software architectures', *Journal of Software Engineering and Applications*, Vol. 10, No. 8, p.677.

Kriouile, A., Addamssiri, N. and Gadi, T. (2015) 'An MDA method for automatic transformation of models from CIM to PIM', *American Journal of Software Engineering and Applications*, Vol. 4, No. 1, pp.1–14.

Leitner, M., Rinderle-Ma, S. and Mangler, J. (2011) 'AW-RBAC: access control in adaptive workflow systems', *in Proceeding of the Sixth International Conference on Availability, Reliability and Security (ARES)*, pp.27–34.

Lengyel, L., Levendovszky, T. and Charaf, H. (2008) 'Validated model transformation-driven software development', *Int. J. Computer Applications in Technology*, Vol. 31, Nos. 1–2, pp.106–119.

Liu, T., Cheng, Y. and Ni, Z. (2012) 'Mining event logs to support workflow resource allocation', *Knowledge-Based Systems*, Elsevier, Vol. 35, No. 1, pp.320–331.

Mejri, A., Ghanouchi, S.A. and Martinho, R. (2015) 'Evaluation of process modeling paradigms enabling flexibility', *Procedia Computer Science*, Vol. 64, No. 1, pp.1043–1050.

Milanovic, M., Gasevic, D. and Rocha, L. (2011) 'Modeling flexible business processes with business rule patterns', *in Proceedings of the 15th Enterprise Distributed Object Computing Conference (EDOC'11)*, pp.65–74.

Mouheb, D., Debbabi, M., Pourzandi, M., Wang, L., Nouh, M., Ziarati, R. and Lima, V. (2015) 'Model-driven architecture and model transformations', in *Aspect-oriented Security Hardening of UML Design Models*, pp.35–45, Springer International Publishing, Cham.

Muller, R., Greiner, U. and Rahm, E. (2004) 'Agent work: a workflow system supporting rule-based workflow adaptation', *Data & Knowledge Engineering*, Elsevier, Vol. 51, No. 2, pp.223–256.

Ngai, E.W.T., Leung, C.H. and Wong, Y.C. (2005) 'Application of the workflow management system in electronic commerce: a case study', *Int. J. Business Information Systems*, Vol. 1, Nos. 1–2, pp.182–198.

Perillo, D. and Di Natale, M. (2017) 'Using MDA to automate the integration of virtual platforms for system-level simulation', *in 2017 IEEE 41st Annual Computer Software and Applications Conference (COMPSAC)*, IEEE, July, Vol. 1, pp.268–277.

Popp, R., Kaindl, H., Nurcan, S., Pimenidis, E., Pastor, O. and Vassiliou, Y. (2014) 'Automated adaptation of business process models through model transformations specifying business rules', *in CAiSE Forum/Doctoral Consortium*, pp.65–72.

Reichert, M. and Dadam, P. (1997) 'A framework for dynamic changes in workflow management systems', *Proceedings of 8th International Workshop on Database and Expert Systems Applications*, France, September, pp.42–48.

Rhazali, Y., Hadi, Y. and Mouloudi, A. (2016) 'Model transformation from CIM to PIM in MDA: from business models defined in DFD to design models defined in UML', *Electronic Journal of Information Technology*, No. 9, pp.1–13.

Selim, G.M., Cordy, J.R. and Dingel, J. (2017) 'How is ATL really used? Language feature use in the ATL zoo', *in Proceedings of the 2017 ACM/IEEE 20th International Conference on Model Driven Engineering Languages and Systems*, IEEE Press, September.

Siemens Nixdorf Informationssysteme AG (1995) *WorkParty User Manual*, Version 2.0.

Sohail, A., Dominic, P.D.D. and Shahzad, K. (2016) 'Business process analysis: a process warehouse-based resource preference evaluation method', *Int. J. Business Information Systems*, Vol. 21, No. 2, pp.137–161.

Suganthi, S. and Nadarajan, R. (2013) 'Role of aspect-oriented approach in dynamic adaptability', *International Journal of Computer Applications in Technology*, Vol. 47, No. 4, pp.334–342.

Sunyé, G. (2017) 'Model consistency for distributed collaborative modeling', *in European Conference on Modelling Foundations and Applications*, Springer, Cham, July, pp.197–212.

Tantan, O.C. and Akoka, J. (2014) 'Automated transformation of business rules into business processes', *in Proceedings of the Twenty-sixth International Conference on Software Engineering and Knowledge Engineering*, pp.684–687.

ter Hofstede, A.H.M., van der Aalst, W.M.P., Adams, M. and Russell, N. (2010) *Modern Business Process Automation: YAWL and its Support Environment*, Springer-Verlag, Berlin, Germany.

Vara, J.M., De Castro, M.V., Del Fabro, M.D. and Marcos, E. (2010) 'Using weaving models to automate model-driven web engineering proposals', *Int. J. Computer Applications in Technology*, Vol. 39, No. 4, pp.245–252.

Weske, M. (1998) 'Flexible modeling and execution of workflow activities', Submitted to *31st Hawaii International Conference on System Sciences (HICSS-31)*.

Workflow Management Coalition Glossary (1996) [online] http://www.aiai.ed.ac.uk/project/wfmc/ARCHIVE/DOCS/glossary/glossary.html (accessed 22 June 2019).

Yang, W. and Xu, X.W. (2011) 'A new approach for integrating process planning with scheduling', *Int. J. Computer Applications in Technology*, Vol. 42, No. 4, pp.253–266.

Zhou, D., Chen, X., Jin, Q., Kuang, Z. and Yang, H. (2017) 'An affinity analysis based CIM-to-PIM transformation', *Multiagent and Grid Systems*, Vol. 13, No. 3, pp.269–286.