Towards the Standardization of Industrial Scientific and Engineering Workflows with QVT Transformations

Corina Abdelahad, Daniel Riesco
Departamento de Informática
Universidad Nacional de San Luis
San Luis, Argentina
{cabdelah, driesco}@unsl.edu.ar

Alessandro Carrara, Carlo Comin, Carlos Kavka
Research and Development Department
ESTECO SpA
Trieste, Italy
{carrara, comin, kavka}@esteco.com

Abstract— **Nowadays, design activities in engineering and many other applied science fields require the execution of computational models in order to simulate experiments. This step is usually automated through the execution of the so-called scientific workflows. A large number of different graphic and execution formats are currently in use today, with no clear signs of convergence into a standard format. Things are different in the area of business processes, where many standards have been defined for both the graphical and the execution representation of business process workflows. Significant efforts are currently being carried out to apply business workflow technology into engineering fields. Nevertheless, one of the main obstacles for the industrial adoption of standards is the large base of existing workflows used currently by industry, which cannot be just thrown away. This paper presents a model-to-model transformation using QVT between a widely used industrial metamodel and the BPMN 2.0 standard metamodel. Legacy workflow support is an essential first step to allow the introduction of the use of a business process standard in scientific and engineering industrial applications.**

Keywords – ***BPMN 2.0; business workflow; industrial workflow; transformation; QVT.***

I. INTRODUCTION

Scientific and industrial design activities depend more and more on the execution of computational models in order to run in-silico experiments. These applications are characterized of being computationally intensive and strongly data-driven. Heavy requirements are imposed, not only on the bare computing technology, but also on the high level execution mechanisms [1][2]. The most widely accepted and effective formalism used to represent these computational processes is in terms of scientific and engineering workflows, which provide a declarative way of stating the required high level specifications. In general terms, a scientific or engineering workflow is an automated business process used to execute complex computational processing tasks [3] in scientific or engineering application areas respectively. These kinds of workflows are widely used in natural science, computational simulations, chemistry, medicine, environmental sciences, engineering, geology, astronomy, automotive industry, aerospace, and other industrial fields. Its use has been extended also to optimization tasks, where the development of complex industrial products is modeled as an optimization cycle which includes an engineering process defined in terms of the collaboration of various engineering services with usually large exchange of information between them [4][5].

It is expected that the success of business process technology in business scenarios can contribute to introduce this already mature technology into the field of scientific and engineering workflows. However, it is not yet the case, even if some interesting contributions are indisputable. The main reason is that scientific and engineering workflows require many features that most business process models do not currently support [6][3]. For example, business workflows usually deal with discrete transactions, but engineering and scientific workflows in most cases deal with many interconnected software tools, large quantities of data with multiple data sources and in multiple formats [7]. Also, engineering services usually have a very long execution duration and depend on the execution environment.

Even if scientific and engineering workflows have been used successfully since many years, most of the tools used to define and execute them are not based on standard technologies. The situation is completely different in the area of business processes, where many well-defined standards have been proposed and are widely used. Some attempts to use a business process standard in the domain of scientific and engineering workflows have been performed, though till now, a single standard cannot be used to represent both the abstract view (used by the engineer to represent the process at the scientific domain) and the workflow representation used for execution (at workflow engine level). However, the last definition of the BPMN standard (the release 2.0) from the Object Management Group (OMG) has been developed with broader objectives, overcoming in fact the limitations that prevented the use of previous versions in scientific and engineering applications [8][9]. From now on, all references with the acronym BPMN are intended as references to version 2.0 of the standard. BPMN defines a formal notation for developing platform-independent business processes, contrasting with specific definitions of business processes such as BPEL4WS (Business Process Execution Language

for Web Services) [10]. BPMN defines an abstract representation for the specification of executable business processes within a company, which can include human intervention, or not. BPMN also allows collaboration between business processes of different organizations. The definition of this new standard allows, for the very first time, to extend the use of workflows from the field of business process to the field of science and engineering.

With BPMN, many companies will be tempted to support a standard workflow for scientific and engineering applications. However, it must be considered that there exists a large base of engineering workflows already designed and used currently by industry, which cannot be just thrown away. In order to provide legacy workflow support, we propose a methodology for the transformation of legacy proprietary workflows into BPMN standard workflows. This approach will provide an extra incentive for companies to abandon proprietary workflows and move to standard technologies coming from the field of business processes. However, the transformation is not without pain. The extra data and process requirements in engineering workflows need to be handled properly. Fortunately, BPMN has been defined with an extension facility which allows to add required constructions without breaking standard compliance.

As a part of the methodology, this paper presents a transformation for selected constructions of a widely used industrial engineering workflow to BPMN, in order to present a valid path to perform legacy workflow conversion to a well-defined standard. It is an extension of the work presented in [1], where the basic methodology was presented. In this present paper, transformations of more complex elements based in BPMN extensions are also considered, providing insights on a not-so-easy to handle BPMN construction, which is essential for the support of scientific and engineering workflows. Also, an extended example is presented, together with a more deep explanation of the legacy workflow model and the results of the transformation in terms of XML elements. New sections were added to present the motivations and a discussion on the proposed approach.

The transformation is defined in QVT, a standard relation language for model transformation defined by the OMG with a specification based on MOF and OCL [11]. The language consents to express a declarative specification of the relationships between MOF models and metamodels supporting complex object pattern matching. A QVT transformation defines the rules by which a set of models can be transformed into a different set [12]. Furthermore, it specifies a set of relations that the elements of the implicated models in the transformation must fulfill. The model types are represented by their corresponding metamodels. A relation in QVT specification consists in a set of transformation rules where a rule contains a source domain and a target domain [13]. A domain is a set of variables to be matched in a typed model, with each domain defining a candidate model and also having its own set of patterns [12]. For more details on QVT, the reader is invited to visit the OMG links [11].

The paper is structured in sections. Section II presents related works. The industrial metamodel used as the source model for transformations is described in Section III. Section IV presents an example of the transformation from the point of view of the workflow designer, while Sections V and VI describe the transformation architecture and the transformation between models, respectively. The paper ends with an example in Section VII, and discussions and conclusions in Sections VIII and IX, respectively.

II. RELATED WORK

The use of scientific and engineering workflows for process automation has been widely analyzed in literature [3]. Many commercial and open source implementations do exist. The most widely used by the open source community are Kepler [14], Triana [15], Taverna [16], Pegasus [17] and KNime [18], with many new frameworks appearing continuously. However, all these scientific and/or engineering workflow frameworks are based in proprietary non-standard formats. In the area of commercial tools, there exists many options like for example modeFRONTIER [4] widely used in CAD/CAE engineering optimization. However, again, all of them are based in proprietary formats.

In [1], the authors present a model-to-model transformation using QVT between a widely used engineering workflow and BPMN 2.0, converting successfully data inputs, input sets and input output specifications into the target format. The approach was validated experimentally in the engineering environment supported by a company in the field of multi-objective optimization. This current paper is an extension of [1].

The use of standards like BPMN 1.0 for the abstract representation of scientific workflows, and BPEL or Pegasus for execution were proposed in the past, but never went too far in industry due to the need to support two different standards for the same workflow [17].

Several works in the field of software engineering are related to the concept of transformation between models, and many of them use BPMN to model business process.

Marcel van Amstel et al. [19] investigate the factors that have an impact on the execution performance of model transformation. This research estimates the performance of a transformation and allows to choose among alternative implementations to obtain the best performance. The results of this study can be used by implementers of transformation engines in order to improve the set of currently available tools.

In this same line, a model-to-model transformation between PICTURE and BPMN 1.0 is presented in [20]. PICTURE is a domain-specific process modeling language for the public administration sector. The transformation allows to model administrative processes in PICTURE and to get BPMN models for these processes automatically, helping electronic government by making possible the implementation of supporting processes. In addition, this research contributes to simplify the development process, improves its flexibility and allows meeting organizational

challenges arising in the development of systems that support electronic government.

In [21], three sets of QVT relations are presented as a mean of implementing transformations in a model-driven method for web development. One of them transforms a high-level input model to an abstract web-specific model. The other two transform the abstract web model to specific web platform models.

In [22], the generation of components of the Java EE 6 business platform from technical business processes modeled with BPMN 1.0 was presented. The generation was obtained by performing three transformations in the context of Model-Driven Architecture, performed with QVT Relations and a MOFScript. This research contributed improving the development productivity and reducing design errors.

A solution for the modeling of Clinical Pathways (CP) processes in terms of standard business process models is presented in [23]. To represent a CP as a process workflow, a high-level semantic mapping between the CP ontology and the BPMN ontology was developed. This research shows how a clinical specific process defined in the CP ontology is mapped to a standard BPMN workflow element. This mapping allows healthcare professionals to model a CP by using familiar modeling constructs. Once ready, they can transform this CP to a business process model and thus leveraging the standard definitions of processes to represent and optimize clinical environments by incorporating process optimization tools.

An example application is presented in [24] to demonstrate an automated transformation of a business process model into a parameterized performance model, thus obtaining significant advantages in terms of easy customization and improved automation.

However, to the best of our knowledge, no other research work has considered BPMN as the target model for transformation in the context of industrial scientific or engineering workflows.

III. ESTECO METAMODEL

The metamodel selected as an example is the workflow model used for modeling simulation workflows by ESTECO, a company specialized in industrial multi-objective optimization[4]. The simulated process is represented with a formalism which provides both a representation for the abstract view (used by the engineer to represent the process) and the associated execution model (used for the real simulation). The abstract view is a human-understandable graphic representation, while the execution model is represented with XML. This last model is used by a workflow engine in order to execute the workflow and perform the simulation.

This workflow, which is typical in this kind of environments, includes one task node for each activity and data nodes used to represent input, output and temporary data objects. Data objects can represent simple data like integer, doubles, vectors, matrices or more complex data like files or databases. Activities correspond to the execution of simulators, scripts and other applications in local or remote locations. Usually, each activity is defined through a set of configuration files, which can be large (many gigabytes being common), and a set of inputs and outputs (which can also be very large files or databases). Distributed execution is required, meaning that the activities specified in the workflow can be executed in different nodes (on the grid or the cloud system[25]), requiring data to be passed between them. More information about the ESTECO metamodel can be found in the documentation provided in the web site [4].

The next sections provide a description of the framework used for the transformation by applying it to a small subset of ESTECO's workflow.

IV. TRANSFORMATION EXAMPLE

As it was mentioned before, the ESTECO and the BPMN notations have both a graphical and an XML representation. Usually, the simulation engineer designs the workflow by using a graphical editor, not being at all interested in the associated XML representation, which is used behind the scenes by the editor and the execution engine as the storage and execution format respectively.

This section presents an example of a transformation from the point of view of the designer, who expects to get a BPMN workflow to be obtained from a previously defined ESTECO workflow as a result of the transformation process. Please note that the example presented in this section is intended to present only data handling aspects, and does not include other components, which also need to be considered when performing a full transformation process.

Figure 1 shows an example of a workflow specified in terms of the ESTECO model. It consists of a sequence of two activities, which performs some computation tasks. Execution starts with the node labeled START, which just transfer the execution flow to the first activity (labeled SUM). This first activity receives two inputs and produces a single output as a result of a computational activity. The second activity (labeled MEAN) takes two inputs, one of them being the output of the previous activity, and produces a single output as a result. The workflow terminates successfully when both tasks are executed properly, reaching the node labeled EXIT, or it can generate an exception reaching the node labeled ERROR.

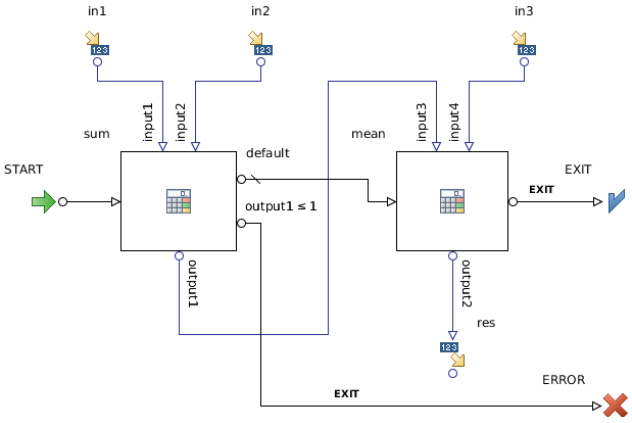

Figure 1. Example of an ESTECO workflow.

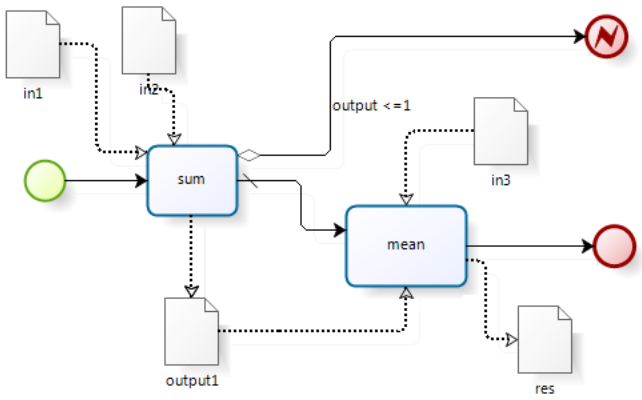

Figure 2. Example of the equivalent BPMN 2.0 workflow.

Figure 2 shows the equivalent BPMN process. Note that the overall graphical structure is not too different between the two workflows. In both of them there is a start node, an end node and an exception event node. There is, however, one extra data object used to transfer intermediate data between the two activities, something that is not required by the ESTECO workflow presented before, which allows direct communication between activities. Note that different kind of arrows and lines are required to indicate the data flow and process flow in BPMN, something that is not so nicely differentiated in the original ESTECO workflow.

An important point to note is that, even if the overall graphical structure is very similar in both workflows, the XML representation is definitely very different. And of course, the transformation process does not take place at the graphical level, but at the XML representation level. This transformation is made possible since both workflow models are defined formally with an XML schema, which provides the basis for a formal transformation process. This transformation process, including selected transformation code in QVT and some examples, is presented in next sections.

V. TRANSFORMATION ARCHITECTURE

Our proposal aims to apply the most recent concepts of business processes to the field of engineering workflows in industrial fields. The use of standards in industry is important since it guarantees portability between tools that support BPMN.

The industrial legacy workflow selected has an XML representation, allowing the use of tools like Medini QVT for transformation [26]. There is no one-to-one correspondence between the different components of ESTECO's workflow and BPMN constructions, since control nodes and data nodes are very differently handled in both models. Also, files and database handling put extra requirements which can only be handled properly with BPMN extensions.

The QVT transformations describe relations between the source metamodel and the target metamodel, both specified in MOF. The transformation defined is then applied to a source model, which is an instance of the ESTECO source metamodel, to obtain a target model, which is an instance of the BPMN target metamodel, as can be seen in Figure 3. The metamodels used in the definition of the transformation are shown at the top level. The specific models to which the transformation defined in the metamodel level is applied in order to obtain BPMN models is shown at the middle level. The lower level represents the instances of the models which can be executed in the corresponding workflow engines.

As mentioned before, activities and processes need data in order to be executed, and in addition, they can produce data during or as a result of their execution. In BPMN, data requirements are captured as *DataInputs* and *InputSets*. The produced data is captured using *DataOutputs* and *OutputSets*. These elements are aggregated in an *InputOutputSpecification* class [2], as can be seen from the UML class diagram presented in Figure 4. The *DataInputs* and *DataOutputs* are additional attributes of the *InputOutputSpecification* element; these elements are optional references to the *DataInputs* and *DataOutputs* respectively. A *DataInput* is a declaration that a particular

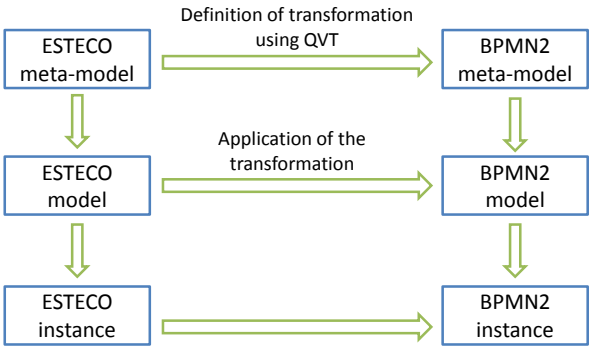

Figure 3. Transformation architecture.

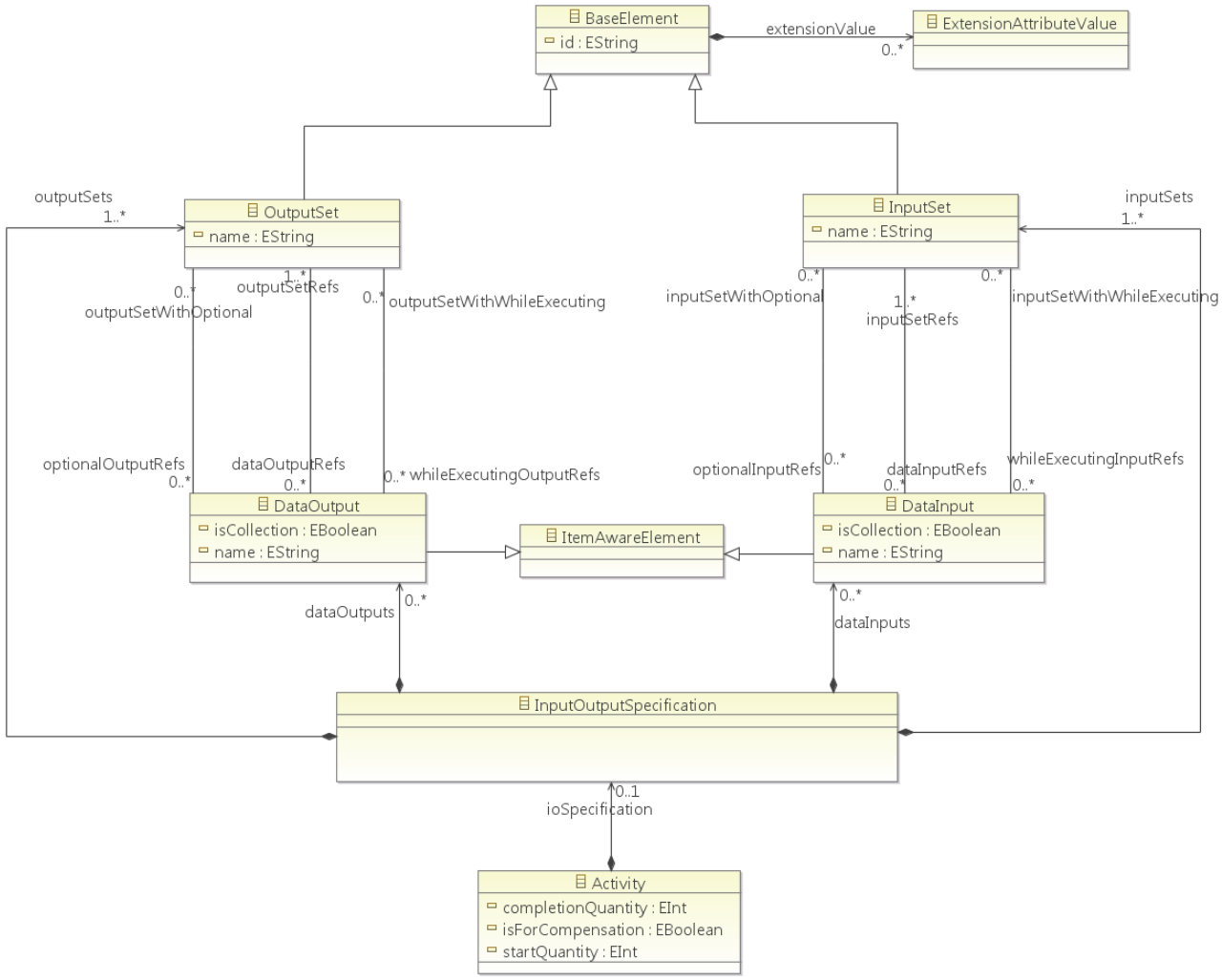

Figure 4. Partial view of the BPMN2 metamodel (from [8]).

kind of data will be used as input of the *InputOutputSpecification*. A *DataOutput* is a declaration that a particular kind of data can be produced as output of the *InputOutputSpecification*. *DataInputs* and *DataOutputs* are *ItemAware* elements. If the *InputOutputSpecification* defines no *DataInput*, it means no data is required to start an Activity. If the *InputOutputSpecification* defines no *DataOutput*, it means no data is required to finish an Activity [8].

The BPMN specification provides an extension mechanism for both the process model and the graphic representation that allows the extension of standard BPMN elements with additional attributes. This mechanism can be used by modelers and modeling tools to add non-standard elements or artifacts to satisfy a specific need. The only requirement is that these extension attributes must not contradict the semantics of any BPMN element [8]. The *ExtensionAttributeValue* class has a relationship with *BaseElement* class, defining a list of attributes or elements that can be attached to any standard BPMN element, as can be seen in Figure 4. As mentioned before, a *DataInput* is an *ItemAwareElement*. All item aware elements inherit the attributes and model associations of *BaseElement*. Therefore, a *DataInput* element inherits the attributes and model associations of *BaseElement*, allowing the extension mechanism to be used by *a DataInput* [8].

A partial view of the ESTECO metamodel with the metaclasses involved in the relations described in this work is shown in the UML class diagram presented in Figure 5. The *TInputDataNode* and *TOutputDataNode* elements inherit the attributes and model associations of *TDataNode*, which in turn, inherits from *TNode*. The *TGeometry* class is the outermost object for all ESTECO elements, i.e., all these elements are contained in a *TGeometry*. The *TInputDataNode* element is a particular kind of *TDataNode* that will be used as input of *TGeometry* to a *Task*. The *TOutputDataNode* element is a particular kind of *TDataNode* which can be produced as output of a *Task* contained in

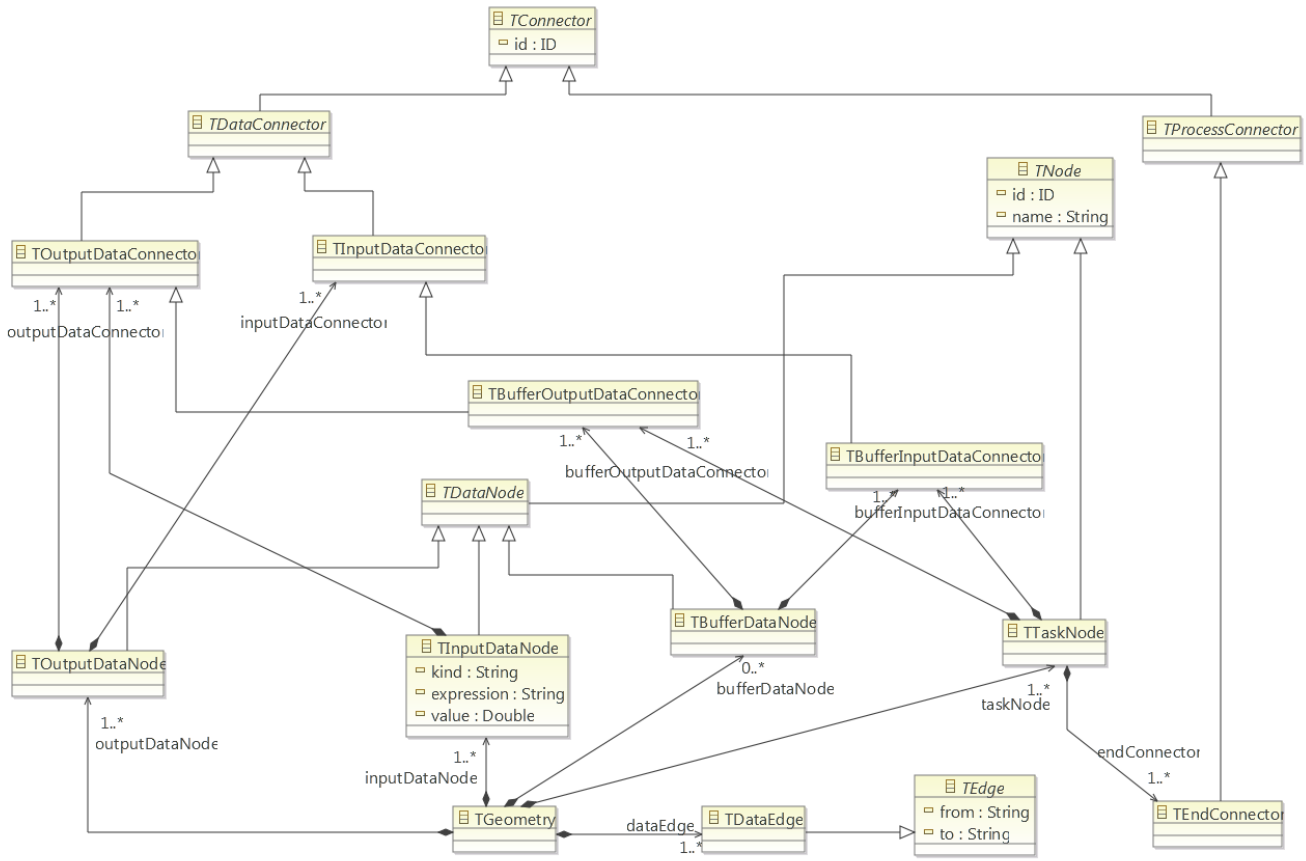

Figure 5. Partial view of ESTECO metamodel (from [4]).

TGeometry. A *TTaskNode* class represents the task that is performed within an industrial workflow.

VI. TRANSFORMATION BETWEEN MODELS

A transformation specifies a group of relations that the elements of the involved models must fulfill. A transformation may have any number of input or output parameters known as domains. For each output parameter, a new model instance is created according to the metamodel of the output metamodel (in this case, the metamodel BPMN).

Each domain identifies a corresponding set of elements defined by means of patterns. A domain pattern can be considered an object template. A relation in QVT defines the transformation rules. A relation implies the existence of classes for each one of its domains. In a relation, a domain is a type that may be the root of a template pattern. A domain implies the existence of a property of the same type in a class. A pattern can be viewed as a set of variables and a set of constraints that model elements must satisfy. A template pattern is a combination of a literal that can match against instances of a class and values for its properties. A domain can be marked as *checkonly* or *enforced*. A *checkonly* domain simply verifies if the model contains a valid correspondence that satisfies the relation. When a domain is *enforced*, if checking fails, the elements of target model can be created, deleted or modified so as to satisfy the relationship.

A relation can contain two sets of predicates identified by a *when* or a *where* clause. The *when* clause specifies the condition that must be satisfied to execute the transformation. The *where* clause specifies the condition that must be satisfied by all model elements involved in the relation, and it may contain any variable involved in the relation and its domains [5]. In the context of transformation, a model type represents the type of the model. A model type is defined by a metamodel and an optional set of constraint expressions. Please note that the definition of these terms can be found in the QVT specification, where the interested reader is referred to [5].

The transformation between ESTECO metamodel and BPMN metamodel is defined as follows:

```
transformation ESTECOToBPMN2(source : esteco_m,
                             target : bpmn2)
```

Note that this transformation takes as input an ESTECO model, which is an instance of the ESTECO metamodel, and produces a BPMN model, that will be an instance of the BPMN metamodel.

Below, the relations which define the mapping between ESTECO metamodel classes and BPMN metamodel classes are presented. This correspondence is not straightforward. As we mentioned in the previous section, the *DataInput*s are captured in *InputSet*s and both are added into an *InputOutputSpecification*. The same happens with the *DataOutput*s. So, in the transformation it is necessary to generate an *IoSpecification* object to aggregate *DataInput*s, *DataOutput*s, *InputSet*s and *OutputSet*s.

The relation used to create an *IoSpecification* object is shown below:

```
relation createIOSpecificationTask {
   checkonly domain source g:esteco_m::TGeometry { };
   enforce domain target t:bpmn2::Task {
   ioSpecification= ioSpecif :
                          bpmn2::InputOutputSpecification {}
  };
  primitive domain id_task:String;
  where {
   getDataInputTask(g,ioSpecif, id_task);
   createInputSetsTask(ioSpecif,ioSpecif);
   getDataOutputTask(g, ioSpecif, id_task);
   createOutputSetsTask(ioSpecif, ioSpecif);
   }
}
```

The relations that are referenced in the previous code, which are used to create InputSets and OutputSets, are the following:

```
relation createInputSetsTask {
  checkonly domain target ioSpecif:
                          bpmn2::InputOutputSpecification {
  };
  enforce domain target ioSpecif :
                          bpmn2::InputOutputSpecification {
       inputSets = input_set :bpmn2::InputSet{
       dataInputRefs= ioSpecif.dataInputs
     }
  };
}

...
```

```
...

relation createOutputSetsTask {
  checkonly domain target ioSpecif:
                          bpmn2::InputOutputSpecification{
  };
  enforce domain target ioSpecif :
                          bpmn2::InputOutputSpecification{
      outputSets = output_set :bpmn2::OutputSet{
      dataOutputRefs= ioSpecif.dataOutputs
    }
  };
```

Note that an *InputSet* is a collection of *DataInput* elements that together define a valid set of data inputs associated to an *InputOutputSpecification*. An *InputOutputSpecification* must define at least one *InputSet* element. An *OutputSet* is a collection of *DataOutput*s elements that together can be produced as output from an Activity. An InputOutputSpecification element must have at least OutputSet element [3].

The relation used to obtain the *DataInputs* of the ESTECO model and the generation of *DataInputs* in BPMN is the following:

```
relation getDataInputTask{
   id_input, name_input : String;
   value_input : Real;
   checkonly domain source g:esteco_m::TGeometry{
     taskNode = t:esteco_m::TTaskNode{
     bufferInputDataConnector = buffer_input :
                    esteco_m::TBufferInputDataConnector {}
     },
     inputDataNode = input : esteco_m::TInputDataNode {
       id = id_input,
       name = name_input,
       value = value_input,
       outputDataConnector = output_data :
                     esteco_m::TOutputDataConnector {}
     },
     dataEdge = data_edge : esteco_m::TDataEdge {}
   };

...
```

```
...

 enforce domain target ioSpecif :
                              bpmn2::InputOutputSpecification  {
        dataInputs = data_input : bpmn2::DataInput {
        id= id_input + '_T',
        name = name_input,
        itemSubjectRef = item : bpmn2::ItemDefinition {
            id = 'DoubleItemDefinition'
        },          extensionValues = extension :
                                        esteco::DocumentRoot{
                default = default : esteco::TDefault {
              simpleValue = simple_value : esteco::TSimpleValue {
                    value = '0'
                }
              }
            }
        }
  };

   primitive domain id_task:String;
   when {
     if (data_edge.from = output_data.id) and
       (data_edge.to = buffer_input.id ) and
       (id_task=t.id) then true else false
  endif;
 }
}
```

Each data input of ESTECO must be transformed into a data input of BPMN. This transformation is straightforward; the QVT code presented before shows the procedure by which the *id*, *name*, *value* and *connectors* are obtained. Note that there is an *extensionValues* attribute referenced in the previous code. This attribute belongs to the *BaseElement* class (Figure 4), which is defined with type *ExtensionAttributeValue*.

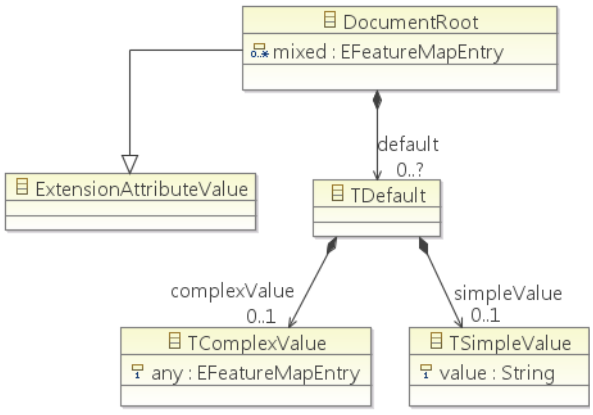

Figure 6. Partial view of the ESTECO XSD definition.

To understand the extensions processing during the transformation process, it is necessary to refer to the definition of types in the ESTECO metamodel. This definition is presented in Figure 6: *DocumentRoot* element inherits the attributes and model associations of *ExtensionAttributeValue*, a class belonging to the BPMN definition, as can be seen in Figure 4. It was necessary to aggregate new attributes: the *Value* attribute, which is contained within *TSimpleValue* and has default value of zero, and the *simpleValue* attribute, which is contained within *TDefault*.

The relation used to obtain the *DataOutput*s of ESTECO model and the generation of *DataOutput*s in BPMN is shown below.

```
relation getDataOutputTask{
   id_output, name_output : String;
   checkonly domain source g:esteco_m::TGeometry {
      taskNode = t:esteco_m::TTaskNode{
      bufferOutputDataConnector = buffer_output :
                    esteco_m::TBufferOutputDataConnector {}
      },
      outputDataNode = output :
                              esteco_m::TOutputDataNode {
        id = id_output, name = name_output,
        inputDataConnector = input_data :
                             esteco_m::TInputDataConnector {}
      },
      dataEdge = data_edge : esteco_m::TDataEdge {}
    };
   enforce domain target ioSpecif :
                             bpmn2::InputOutputSpecification  {
      dataOutputs = data_output : bpmn2::DataOutput {
      id= id_output + '_T',
      name = name_output,
      itemSubjectRef = item : bpmn2::ItemDefinition {
        id = 'DoubleItemDefinition'      }
      }
    };
    primitive domain id_task:String;
    when {
      if (data_edge.from = buffer_output.id) and
        (data_edge.to = input_data.id )  and
        (id_task=t.id) then  true else false
      endif;
   }
}
```

VII. A TRANSFORMATION EXPERIMENT

This section presents an example of a transformation by using the QVT code presented before. The QVT transformations were defined by using Medini QVT, a tool developed by IKV++ technologies with an Eclipse

integration [26]. Medini QVT allows both single direction and bidirectional transformations. The core engine implements the OMG's QVT Relations standard, and is licensed under EPL (Eclipse Public License). The Relations language implicitly creates trace classes and objects to record the events that occurred during a transformation execution.

The QVT Process package contains classes that are used for modeling the flow of Activities, Events, and Gateways, and their sequence within a Process. When a Process is defined it is contained within Definitions [8]. A Process is instantiated when one of its *Start Events* occurs. The *End Event* indicates where a Process will end, finishing the flow of the Process. Data requirements and Data Outputs are contained within an *ioSpecification* object, which defines not only the inputs and outputs, but also the *InputSets* and *OutputSets* for the Process and the Activities [8].

Figure 7 presents the results of the execution of a transformation when applied to one single activity node in the workflow defined in Figure 1. Each box corresponds to an XML element, and the hierarchy between the elements is represented with the tree-like structure. Each task has its own *ioSpecification* object, which contains its own data. Hence, the transformation generates an *ioSpecification* object to combine *DataInputs*, *DataOutputs*, *InputSets* and *OutputSets*, as it was mentioned previously.

Each data input of the ESTECO workflow task is captured as an *inputDataNode* object, which is transformed into a *dataInput* of BPMN. To satisfy specific needs of ESTECO, it has become necessary to use the extension mechanism of BPMN for *DataInput* handling. As it was shown in the previous section, the QVT code for the *getDataInputTask* relation presents the procedure by which the *id*, *name*, *value* and *connectors* are obtained and the

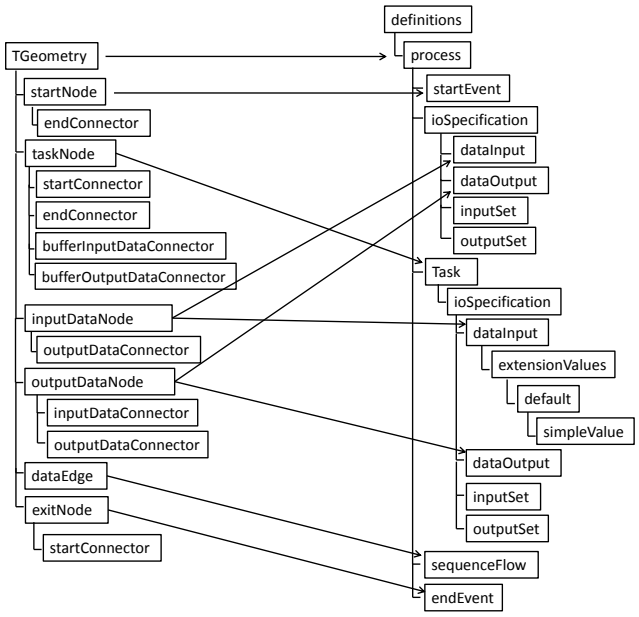

Figure 7. Correspondence between the XML elements during a transformation by considering a single activity from the workflows defined in Figures 1 and 2.

extensionValues element is generated. The two new elements contained in the *extensionValues* element are *default* and *simpleValue*.

Each data output of the activity node is captured as an *outputDataNode* object, which is in turn transformed into a *dataOutput* of BPMN. This transformation has been presented in the definition of the relation *getDataOutputTask* introduced before. Note that an *InputOutputSpecification* must define at least one *InputSet* element and at least one *OutputSet* element. Once the data input and output have been generated, the *inputSet* and *outputSet* are in turn generated. The corresponding QVT generation code can be found in the relations *createInputSetsTask* and *createOutputSetsTask* respectively.

VIII. DISCUSSION

The paper has proposed the use of a standard model-to-model transformation technology in order to convert scientific and engineering workflows into a business process standard format. The main contributions of the proposal are the following:

- **Technical feasibility**: the paper has shown that QVT provides an effective and standard method to transform scientific and engineering workflows into a standard business process format. It has shown that concepts coming from model driven architecture (MDA) can be applied in the domain of science and engineering design. Being QVT part of the OMG standards, these concepts can be useful as the basis for the development of domain-specific Model Driven Engineering tools [27].
- **Incentive to support standards in scientific and engineering community**: companies that use a proprietary workflow format that is properly defined with a model schema, can use a similar transformation process to export their workflows into a standard format. There are no restrictions on the use of QVT for this purpose, since it is an open standard defined by the OMG with many alternative implementations available.
- **Transformation example with a real workflow format widely used in industry**: the legacy workflow model is a widely used format in engineering all around the world, definitely not a model defined just for this paper evaluation. ESTECO is a world-wide leader in the domain of multi-objective optimization applied to engineering design, which is currently pursuing strong efforts to increase support for standards in the multi-objective optimization domain in the context of engineering processes, as it can be seen in [5].

Note that the example presented in this paper is intentionally small, in order to effectively demonstrate the approach, without introducing the reader into extra complexity generated by a larger example. Due to this successful results, the company plan to extended it to support the full specifications of the original legacy workflows,

becoming a part of the tool sets provided in a new BPMN compatible workflow environment.

IX. CONCLUSION

The paper has proposed the use of QVT-based transformation technology in order to transform engineering workflows defined in a legacy proprietary format to a well-defined business process standard. An example involving only data related components has been presented. The approach has been validated experimentally in an engineering environment supported by a company specialized in multi-objective optimization. It is important to stress that this transformation allows the conversion of most ESTECO industrial workflows to BPMN, consenting their execution in BPMN workflow engines with adequate extensions support.

The objective of this work has been to apply important concepts of business processes to the industrial field. Furthermore, it intended to show the importance of the use of standards in industrial fields in order to guarantee portability between tools that support BPMN. As a more general objective, it is expected that the use of a standard for scientific and engineering workflows will facilitate the collaboration between scientists and industrial designers, enhance the interaction between different engineering and scientific fields, providing also a common vocabulary in scientific and engineering publications [5].

ACKNOWLEDGMENT

The authors thank the reviewers of the ICSEA'12 conference and the IARIA Journal for the very useful comments that have contributed to enhance both the original and the extended versions of the paper.

REFERENCES

[1] Corina Abdelahad, Daniel Riesco, Carlo Comin, Alessandro Carrara, and Carlos Kavka, "Data Transformations using QVT between Industrial Workflows and Business Models in BPMN", Proceedings of the Seventh International Conference on Software Engineering Advances ICSEA, 2012, IARIA.

[2] Yolanda Gil, Ewa Deelman, Mark Ellisman, Thomas Fahringer, Geoffrey Fox et al., "Examining the challenges of scientific workflows", IEEE Computer vol. 40, no. 12, 2007, pp. 24-32.

[3] Cui Lin, Shiyong Lu, Xubo Fei, Artem Chebotko, Darshan Pai et al., "A reference architecture for scientific workflow management systems and the VIEW SOA solution", IEEE Transactions on Service Computing, vol. 2, no. 1, 2009, pp. 79-92.

[4] ESTECO S.p.A., "modeFRONTIER applications across industrial sectors involving advanced CAD/CAE packages", http://www.esteco.com/home/mode_frontier/by_industry, [retrieved: March, 2013]

[5] Carlo Comin, Luka Onesti, and Carlos Kavka, "Towards a Standard Approach for Optimization in Science and Engineering", Proceedings of the 8th International Conference on Software Engineering and Applications ICSOFT-EA, 2013, SciTePress.

[6] Li Hongbiao, Li Feng, and Yu Wanjun, "The research of scientific workflow engine", Proceedings of the IEEE International Conference on Software Engineering and Service Sciences (ICSESS), 2010, pp. 412-414.

[7] Shown Bowers. "Scientific Workflow, Provenance and Data Modeling Challenges and Approaches", Journal on Data Semantics, vol. 1, pp. 19-30, 2012, Springer.

[8] Object Management Group (OMG), "Business process model and notation", http://www.omg.org/spec/BPMN/2.0 [retrieved: March, 2013]

[9] The Business Process Management Initiative (BPMI.org), http://www.bpmi.org/ [retrieved: October, 2012]

[10] Oasis, "Web services business process execution language version 2.0", http://docs.oasis-open.org/wsbpel/2.0/wsbpel-v2.0.html, [retrieved: March, 2013]

[11] Object Management Group (OMG), "Modeling and metadata specifications", http://www.omg.org/spec, [retrieved: October 2012]

[12] Object Management Group (OMG), "Meta object facility (MOF) 2.0 query/view/transformation, V1.1", http://www.omg.org/spec/QVT/1.1 [retrieved: October, 2012]

[13] Li Dan, "QVT based model transformation from sequence diagram to CSP", Proceedings of the 15th IEEE International Conference on Engineering of Complex Computer Systems (ICECCS), 2010, pp. 349-354.

[14] Bertram Ludascher, Ilkay Altintas, Shawn Bowers, Julian Cummings, Terence Critchlow et al., "Scientific Process Automation and Workflow Management", in "Scientific Data Management: Challenges, Technology, and Deployment", edited by A. Shoshan and D. Rotem, 2009, Chapman and Hall/CDC.

[15] Ian Taylor, Matthew Shields, Ian Wang, and Andrew Harrison, "The Triana workflow environment: architecture and applications", in "Workflows for e-Science: Scientific Workflows for Grids", I. Taylor et al., 2007, Springer.

[16] Pablo Missier, Stian Soiland-Reyes, Stuart Owen, Wei Tan, Alexrandra Nenadic et al., "Taverna, Reloaded", Lecture Notes in Computer Science, vol. 6187, pp. 471-481, 2010, Springer.

[17] Mirko Sonntag, Dimka Karastoyanova, and Ewa Deelman, "Bridging The Gap Between Business And Scientific Workflows", Proceedings of the ESCIENCE 2010, 6th IEEE International Conference on e-Science, 2010, IEEE Computer Society.

[18] Michael Berthold, Nicolas Cebtron, Fabian Dill, Thomas R. Gabriel, Tobias Kotter et al., "KNIME: The Konstanz Information Miner", in "Data Analysis, Machine Learning and Applications", ed. H. Bock, W. Gaul, M. Vichi, pp. 319-326, 2008, Springer.

[19] Marcel van Amstel, Steven Bosems, Ivan Kurtev, and Luís Ferreira Pires, "Performance in model transformations: experiments with ATL and QVT", Lecture Notes in Computer Science, Volume 6707, Theory and Practice of Model Transformations, Springer, 2011, pp. 198-212.

[20] Henning Heitkoetter, "Transforming PICTURE to BPMN 2.0 as part of the model-driven development of electronic government systems", Proceedings of the 44th Hawaii International Conference on System Sciences (HICSS), 2011, pp. 1-10.

[21] Ali Fatolahi, Stéphane Somé, and TimothyLethbridge, "Automated generation of abstract web models using QVT relations", Technical Report TR-2010-06, School of Information Technology and Engineering, University of Ottawa, September 2010.

[22] Narayan Debnath, Carlos Alejandro Martinez, Fabio Zorzan, Daniel Riesco, and German Montejano, "Transformation of business process models BPMN 2.0 into components of the Java business platform", Proceedings of the Industrial Informatics (INDIN), 10th IEEE International Conference on Digital Objects, 2012, pp. 1035-1040, IEEE

[23] Nima Hashemian and Samina Sibte Raza Abidi, "Modeling clinical workflows using business process modeling notation", Computer-Based Medical Systems (CBMS), 25th International Symposium on Digital Object, 2012 , pp. 1-4, IEEE

[24] Paolo Bocciarelli and Andrea D'Ambrogio, "A BPMN extension for modeling non functional properties of business processes", Proceedings of the 2011 Symposium on Theory of Modeling & Simulation: DEVS Integrative M&S Symposium, Springer-Verlag, 2011, pp. 160-168.

[25] Gideon Juve and Ewa Deelman, "Scientific workflows and clouds", ACM Crossroads, vol. 16, no. 3, 2010, pp. 14-18.

[26] IKV, "The Medini QVT project", http://projects.ikv.de/qvt, [retrieved: March, 2013]

[27] D. Schmidt, "Guest Editor's Introduction: Model-Driven Engineering", Computer IEEE, vol. 39, pp. 25-31, 2006.