Transformations from SPEM Work Sequences to BPMN Sequence Flows for the Automation of Software Development Process

M. Perez Kota, D. Riesco, N. Debnath, G. Montejano
1Universidad Nacional de San Luis – Argentina, {driesco, gmonte}@uns.edu.ar
2Department of Computer Science – Winona State University – MN 55987 – USA, ndebnath@winona.edu
3Universidad de Vigo, Campus As Lagoas-Marcosende, Vigo, Pontevedra, España, mpcota@uvigo.es

Abstract

Generally, different companies use distinct software development process. Due to it, OMG has specified an architecture to can define a software development process. This architecture uses a metamodel which is able to be abstracted from particular characteristics and to give the possibility of defining different software development processes. The Software Process Engineering Metamodel (SPEM) is a metamodel used for describing a concrete software development process. Also, the OMG specify a standardized language to model business processes. This language is called Business Process Modeling Notation (BPMN).

In this paper, a formalization of transformations between SPEM and BPMN is specified. The transformations are done using the language of formal specification RSL.

One time that we obtain a BPMN specification, it is possible to generate XPDL code or BPEL code. The workflow technology can use this code and execute the process. It allows automate the business process, but in this case, the process is a software development process. Consider that workflow technologies automate, totally or partially, the business processes enabling an interaction between men and machines, under an established set of procedural rules. Such technologies use standard languages, like BPEL or XPDL, to specify the business process which is executed in a workflow engine. The BPMN allows to generate BPEL or XPDL codes. Therefore, if we specify a software development process using SPEM, we will automate this process using the workflow technology due to transformations between SPEM and BPMN done in this paper.

Key Words— BPMN, RSL, SPEM, Workflow.

1. INTRODUCTION

Businesses processes are groups of tasks related logically run to obtain a business result. Business processes can be controlled and manager by a software based system. The workflow technology automates business processes. This automation results in an important empowerment of these process virtues. Improvements in the organization’s performance, efficiency and productivity are obtained.

The workflow paradigm offers interoperability with other systems, implementation in diverse environments, easiness for the management and monitoring of human resources [10].

The case of software development industry is not different from the rest of the industries. Within this industry, business processes are found which are designed for the construction or generation of a good quality (software) product in a given time. The most important business product within the software development industry is known as “development methodologies”, in charge of guiding production. Nowadays, software engineers work to optimize the development processes. Software engineering tool developers can take advantage from the connection between the management of software development processes and the workflow [17].

A workflow is defined as the total or partial automation of a business process, during which documents, information or tasks are interchanged among the participants according to a pre-established set of procedural rules [10]. Due to the great diversity of workflow products it has been necessary to have a common language that allows the interoperability between these products [1]. The Workflow Management Coalition (WFMC) has standardized the business process automation. The standard proposed by the WFMC includes a workflow metamodel and a language of specification XPDL (XML Process Definition Language) based on this metamodel [2].

A workflow engine is a software system that controls the execution of the activities defined in the workflow. The WFMC has defined a Workflow Reference Model. This model defines 5 interfaces for the interoperability of different products with a workflow engine.


The OMG has proposed a standardized notation which supports several workflow process languages, including XPDL, called BPMN [3]. The BPMN has translations into different languages as XPDL and BPEL4WS code [4].

BPMN defines a notation for the definition of business processes, which is an independent platform compared to specific definitions (for example XPDL or BPEL) of business processes. This notation defines an abstract representation for the specification of business processes executed within a company/organization. Departing from a BPMN model the definition of a business process in a specific language can be obtained through a transformation. In [3] the correspondence is defined between BPMN and BPEL. The elements of the notation are specified in the BPMN metamodel [9].

The OMG defined, not only the BPMN language, also a
Software Process Engineering Metamodel (SPEM) [5]. SPEM is a metamodel capable to be abstracted from particular characteristics and to give the possibility of defining the different software development processes.

RAISE (RAISE Specification Language) is a language of formal specification of RAISE Method [6]. It provides a math-based notation, which is useful to specify, design and develop software formally. RSL permits to specify abstractions, systems with sequential specifications and also concurrent systems, besides, systems of a great size to be modularized, and the splitting into subsystems that will be developed separately. RSL allows the low level operational design to be expressed in detail from which the final code extraction is performed. It also allows the construction from the specification to the design using only one formalism.

RAISE stands for Rigorous Approach to Industrial Software Engineering [7]. The objective of RAISE is to develop a notation, techniques and tools that enable the industrial use of a formal method in the creation of software systems. RAISE includes a great number of techniques, strategies and tools to perform the development and trials of systems [8].

Our research aims to set the formal bases to automate a software development process by the automation of business processes using workflow technology.

To achieve this automation, a translation is proposed for software development processes specified in SPEM into a specification of workflow processes based on the BPMN standard. To do so, first formalization is obtained using RSL language of the SPEM metamodel of work breakdown elements including work sequence. Then the BPMN metamodel of activities and its sequence flows are formalized in RSL language. And finally, the axioms that allow to relate both metamodels is specified in RSL.

The structure of this research work is shown below. Section 2 explains the metamodel SPEM. Section 3 shows the general structure of the automation. Section 4 describes the formalization of SPEM work breakdown element. Section 5 specifies the formalization of SPEM work sequence. Section 6 shows the formalization of BPMN activity. Section 7 specifies the formalization of BPMN sequence flow. Section 8 and 9 specify the formalization of the semantics between SPEM and BPMN. Section 10 describes the conclusions.

2. SPEM

To specify the activities proposed by a particular development process, the OMG defined a Software Process Engineering Metamodel [5].

Generically, for the definition of new languages, the OMG defines an architecture based on four levels of abstraction that allow to distinguish the different concept levels that take part in the modelling of a system. These levels are called M0, M1, M2 and M3.

SPEM describes a generic metamodel for the description of specific software processes. This metamodel works as a template for the creation of models of concrete processes, such as the Rational Software Development Unified Process (RUP) or the model for assessment and improvement of ISO 15504 processes. Therefore, SPEM is a metamodel of level M2 by MOF [16], whereas RUP or ISP 15504 are defined according to SPEM in level M1.

3. RSL to specify the Management Automation.

The specifications developed in the following sections have been verified by tools provided by the International Institute for Software Technology of the United Nations University (IIST/UNU) [8].

The sections contain the formal definition of the work breakdown element of the metamodel for software development processes, based on the SPEM 2 model [5], using the RSL formal language of RAISE method.

Concepts related to SPEM work breakdown element are presented through their metamodel, as RSL schemes, where each scheme is the definition of the concepts required to define a software development process.

This formalized standard structure permits to establish the correspondence that leads us to the automation of the management by means of languages of business process execution. These languages are reached through the translation of a model represented in BPMN. This is why it is necessary to formalize the SPEM metamodel, then the BPMN metamodel, and finally to establish the formalization of the relation between both schemes.

The general structure of the formalization of SPEM and BPMN concepts and its transformation may be divided into three parts:

- Types: it specifies the type of the object being defined.
- Functions: it specifies the functions and characteristics of the object being defined.
- Axioms: it specifies the general restrictions that should be complied with to define its semantics.

4. Formalization of SPEM Work Breakdown Element

A work break down element is a special breakdown element that provides “specific” properties for breakdown element which represent work [5]. It establishes the order that the work break down elements have along their operational sequences.

```object EP_BREAKDOWNELEMENT :
  with TYPES in
  class type
    EP_BreakdownElement =
      EP_BREAKDOWNELEMENT_EP_BreakdownElement,
    EP_WorkBreakdownElement =
      |o: EP_BreakdownElement • ```
is_an_EP_WorkBreakdownElement(o)

Value

// isRepeatable is used for defining work repetitions. Iterations of a work breakdown element that isRepeatable, will be repeated more than once in the same set of devices. For example, if there is the need for modelling a specific activity instance, which represents an iteration in progress, then this function will turn true indicating that the activity and all the sub-activities will be performed more than once.

/*
isRepeatable: EP_WorkBreakdownElement → Bool,

*/

// isOngoing indicates that if this is true for an instance of a work breakdown element, then the element describes a long piece of work without a fixed duration or status end. For example the work breakdown element can represent the administrator’s work who continuously operates in order to ensure that the system keeps a certain state. */

isOngoing: EP_WorkBreakdownElement → Bool,

/* The function linkToPredecessor relates a work breakdown element with its predecessor. Each work breakdown element can have information associated with the predecessor. Such information is represented in instances of the work sequence metaelement that defines the type of predecessor. */

linkToPredecessor: EP_WorkBreakdownElement →
EP_WorkSequence_Id-set.

// The function linkToSuccessor relates a work breakdown element with its successor. Each work breakdown element can have the information associated to the successor which is represented in instances of the work sequence metaclass defining the type of predecessor. */

linkToSuccessor: EP_WorkBreakdownElement →
EP_WorkSequence_Id-set,

/* . . . this is a part of the specification */

end

Further on there is an explanation of the group of Work Breakdown element:

EP_BREAKDOWNELEMENTS_,
EP_WORKBREAKDOWNELEMENT_

object EP_WORKBREAKDOWNELEMENT_: with TYPES in
class

type
EP_WorkBreakdownElement =

*/

Further on there is an explanation of the group of Work Breakdown element:

EP_BREAKDOWNELEMENTS_,
EP_WORKBREAKDOWNELEMENT_

object EP_WORKBREAKDOWNELEMENT_: with TYPES in
class

type
EP_WorkBreakdownElement =

EP_WORKBREAKDOWNELEMENT_
EP_WorkBreakdownElement,
EP_WorkBreakdownElements = {}
super:EP_BREAKDOWNELEMENTS_,
EP_BreakdownElements::
(∀ id: EP_BreakdownElement_Id •
id isin dom super
=>EP_WORKBREAKDOWNELEMENT_
is_an_EP_WorkBreakdownElement(sup(id)))

Value

empty:EP_WorkBreakdownElements = [],

add: EP_WorkBreakdownElement_Id ×
EP_WorkBreakdownElement ×
EP_WorkBreakdownElements ↦
EP_WorkBreakdownElements
add(id, o, c) is c !! [id => o]
pre ~ is_in(id, c),

del: EP_WorkBreakdownElement_Id ×
EP_WorkBreakdownElements ↦
EP_WorkBreakdownElements
del(id, c) is c \ {id}
pre is_in(id, c),

is_in: EP_WorkBreakdownElement_Id ×
EP_WorkBreakdownElements → Bool
is_in(id, c) is id isin dom c,

get: EP_WorkBreakdownElement_Id ×
EP_WorkBreakdownElements ↦
EP_WorkBreakdownElement
get(id, c) is c(id)
pre is_in(id, c),

consistent:EP_WorkBreakdownElements → Bool
consistent(c) is
(∀ id: EP_WorkBreakdownElement_Id •
id isin dom c
=> EP_WORKBREAKDOWNELEMENT_.consistent(c(id)))

/* . . . this is a part of the specification */

end

5. Formalization of SPEM Work Sequence

Work sequence is a break down element. It represents a relation between two work break down elements where one of them depend on the other to begin the execution. The work sequence defines relations of predecessor and successor among work break down elements.
Processes are a set of activities. They are described by subprocesses and tasks. */
with TYPES in
class
type
FlowObject = FLOWOBJECT_.FlowObject,
Activity = {o: FlowObject • is_an_Activity(o)}
value

/* This is the number of token that must arrive from a simple sequence flow before the activity starts */
beginingQuantity: Activity → Int

/* The type of activity could be Subprocess or Task */
activityType: Activity → activityType,

/* The state of an activity is determind when it is being executed by a process engine. The state of an activity can be used in assignment expressions. */
state: Activity → ActivityState,

/* This could be of two different types, standard instance or multiple instance */
cycleType: Activity → activityCycleType,

/* Set of properties which are added by the modeler, which can be attached to the activity and are local. Cardinality: 0..n for each Activity. */
properties: Activity → Property_Id-set,

/* An output is one or more devices that can be connected to another activity by means of an association */
outputs: Activity → Output_Id-list,

/* An input is a device that can be connected with other activities by means of associations. */
inputs: Activity → Input_Id-list,

/* It defines the number of token that are generated by the activity. */
finishedQuantity: Activity → Int

axiom

/* The number of token that must arrive from the simple sequence flow before the activity starts should be greater than 1 */
beginingQuantity as cc post cc >= 1,

/* . . . this is a part of the specification */
end

The following specification allows to define the set of activities of the object 'Activity' defined above. It is defined as a map whose domain is a set of activity identifiers and its

6. Formalization of the BPMN Activity.

To formalize the relations between SPEM and BPMN, it is necessary to formally define the BPMN concepts related to the SPEM work breakdown element.

The most generic concept in BPMN is the activity. It is a task executed by a business process. An activity can be atomic or non-atomic (compound). The types of activities that are parts of the standard specification are: subprocess and task [3].

The formal specification of Activity, which is presented in an unfinished form due to lack of space, is as follows:

FLOWOBJECT_

object ACTIVITY_ :

/* Activity: it refers to work performed within a process. Activities can be atomic or compound. Among the activities there are two kinds called subprocess and task. */

/* . . . */
end
range is the set of activities. It relates an activity identifier with an activity. It has the necessary functions incorporated to handle the map.

FLOWOBJECTS_. ACTIVITY_

object ACTIVITIES_ :
with TYPES in
  class
    type
      Activity = ACTIVITY_.Activity,
      Activities = {\{super:FLOWOBJECTS_.FlowObjects •
        (\forall id: FlowObject_Id • id ∈ dom super
        ⇒ ACTIVITY_.is_an_Activity(super(id))))\} }
  value
    empty: Activities = [],
  add: Activity_Id × Activity × Activities ⊆ Activities
    add(id, o, c) ⇐ c ![id +> o]
    pre is_in(id, c),
  del: Activity_Id × Activities ⊆ Activities
    del(id, c) ⇐ c \{id\}
    pre is_in(id, c),
  is_in: Activity_Id × Activities → Bool
    is_in(id, c) ⇐ \{id\} \subseteq dom c,
  get: Activity_Id × Activities → Activity
    get(id, c) ⇐ c(id)
    pre is_in(id, c),
  consistent: Activities → Bool
    consistent(c) ⇐ (\forall id: Activity_Id • id ∈ dom c
    ⇒ ACTIVITY_.consistent(c(id))))
    /* . . . this is a part of the specification */
end

8. Semantic formalization between SPEM workBreakdownElement and BPMN Activity.

In SPEM, a work break down element is an abstract concept of work which provides the specific properties for breakdown structures.

In BPMN, the concept of a work activity is work executed by a business process. An activity can be atomic or non-atomic (compound). The types of activities that are part of the standard specification are: subprocess and task. These two subtypes are related to other specific concepts of SPEM. Therefore it is understood that a process is a set of activities, so it is the SPEM with the work break down elements.

In SPEM, among the work break down elements, there are the activities and tasks, both corresponding to atomic and non-atomic break down elements. This correspondence between the atomic and non-atomic elements of SPEM with BPMN is established in axioms.

The formalization of the correspondence between the work break down element of SPEM and the activity of BMPN is as follows:

```
../SPEM2/EP_WORKBREAKDOWNELEMENTS_.
../SPEM2/TYPES,
../BPMN/ACTIVITIES_
```

```
object WorkBreakdownElement_Activity_ :
  with TYPES in
```
class
  value
  tWorkBreakdownElement:
    EP_WORKBREAKDOWNELEMENT_.
    EP_WorkBreakdownElement
    \rightarrow ACTIVITY_.Activity,

  tWorkBreakdownElementID:
    EP_WorkBreakdownElement_Id \rightarrow
    Activity_Id,

  tWorkBreakdownElements:
    EP_WORKBREAKDOWNELEMENTS_.
    EP_WorkBreakdownElements
    \rightarrow ACTIVITIES_.Activities

axiom

  -- Axiom 1
  tWorkbreakdownElements(EP_WORKBREAKDOWNELEMENTS_.empty)
  \equiv ACTIVITIES_.empty,

  -- Axiom 2
  \forall e:EP_WORKBREAKDOWNELEMENTS_.
  EP_WorkBreakdownElements,
  id:EP_WorkBreakdownElement_Id •
  tWorkBreakdownElements(e) \equiv
  ACTIVITIES_.add(\n    tWorkBreakdownElementID(id),\n    tWorkBreakdownElement(EP_WORKBREAKDOWNELEMENTS_.get(id, e)),\n    tWorkBreakdownElements(EP_WORKBREAKDOWNELEMENTS_.
      del(id, e)))\n  pre EP_WORKBREAKDOWNELEMENTS_.
    is_in(id, e),

  -- Axiom 3
  \forall e:EP_WORKBREAKDOWNELEMENT_.
  EP_WorkBreakdownElement •
  EP_WORKBREAKDOWNELEMENT_.isRepeatable(e)
  \Rightarrow
    ACTIVITY_.cycleType(\n      tWorkBreakdownElement(e))
    = cycleMultipleOccurences
  \lor
    \sim EP_WORKBREAKDOWNELEMENT_.
    isRepeatable(e)
  \Rightarrow
    ACTIVITY_.cycleType(\n      tWorkBreakdownElement(e))
    = cycleStandard,

  -- Axiom 4
  \forall e:EP_WORKBREAKDOWNELEMENT_.
  EP_WorkBreakdownElement •
  \exists pid:Property_Id •
  EP_WORKBREAKDOWNELEMENT_.
    isOngoing(e)
  \Rightarrow
    pid \in ACTIVITY_.properties(\n      tWorkBreakdownElement(e))
  \land
    isOngoing(pid)
end

The general basic structure, axioms 1 and 2, is to have functions that transform:

- a work break down element of SPEM into an activity of BPMN,
- an identifier of a SPEM work break down element into an identifier of a BPMN activity, and
- a function for which each work break down element of SPEM searches its correspondence with an activity of BPMN.

These two axioms set the generic correspondence without specifying restrictions to which the BPMN elements should comply with so that they are semantically equivalent to the correspondent SPEM elements. This restriction semantics is set by the rest of the axioms.

Axiom 3 isRepeatable is used for defining work repetitions. The iterations of a work break down element that isRepeatable, will be repeated more than once in the same set of devices. For example if it is necessary to model an instance of a specific activity, which represents an iteration in a process, then this function will turn true indicating that the activity and all the subactivities will be performed more than once.

So, if a work break down element is repeatable (function with true return) the corresponding activity will have a cycle type of multiple instance. Otherwise it will be a standard cycle. This semantics is established by the axiom 3.

Axiom 4 specifies that the characteristic of a work break down element is true if it is long, then the element describes a long piece of work without a fixed duration or end of state. It could represent work of an manager that is continuously working to ensure that the system keeps a given state. This characteristic is assured by this axiom whith a property in BPMN activity where it is specified that such activity must be long.

This way the correspondence of the functions of a work break down element is specified by associating them with the work sequence, linkToPredecessor and linkToSuccessor, which are related to BPMN Flow Sequence.
9. Semantic of SPEM Work Sequence in BPMN.

The work sequences represent relations between two work break down element of SPEM where one of them depend on the other work break down element for the beginning of its execution.

The work break down elements of SPEM are directly related with BPMN activities, such as defined in previous sections. These activities are ordered through sequence flows. These flows show a source and a destination, both BPMN activities.

The SPEM work sequence is formalized through to BPMN sequence flow. In this section, the general scheme is showed. Other axioms have been defined but they can not written in this section due to space. In the following paragraphs, the axioms are written separated for a better comprehension.

axiom

\[ \forall \, st : \text{EP\_WORKSEQUENCE\_EP\_WorkSequence}, \]
\[ eId1 : \text{EP\_WorkBreakdownElement\_Id}, \]
\[ eId2 : \text{EP\_WorkBreakdownElement\_Id} \cdot \]
\[ \exists \, fs : \text{SEQUENCEFLOW\_SequenceFlow}, \]
\[ aId1 : \text{Activity\_Id}, aId2 : \text{Activity\_Id} \cdot \]
\[ \text{EP\_WORKSEQUENCE\_linkToPredecessor(st)} = eId1 \]
\[ \wedge \]
\[ \text{GRAPHICCONNECTIONOBJECT\_source(fs)} = aId1 \]
\[ \wedge \]
\[ \text{WorkBreakdownElement\_Activity\_}, \]
\[ t\text{WorkBreakdownElement}\text{\_ID(eId1)} = aId1 \]
\[ \wedge \]
\[ \text{EP\_WORKSEQUENCE\_linkToSuccessor(st)} = eId2 \]
\[ \wedge \]
\[ \text{GRAPHICCONNECTIONOBJECT\_destination(fs)} = aId2 \]
\[ \wedge \]
\[ \text{WorkBreakdownElement\_Activity\_}, \]
\[ t\text{WorkBreakdownElement}\text{\_ID(eId2)} = aId2 \]
\[ \Rightarrow \]
\[ t\text{WorkSequence}(st) = fs, \]

The figure 1 shows a SPEM work sequence is related to BPMN sequence flow through two functions with respect to the source of its elements: the function linkToPredecessor of the work sequence and the function source of sequence flow. The same occurs with the destination of the elements through functions linkToSuccessor and destination. Consider that the links to predecessor are work break down elements of SPEM and the source of BPMN sequence flow is an activity.

8. Conclusion

In this paper, formalization of the process of transformation using RSL is done. The formalization in RSL of the software development process based on the SPEM metamodel of work breakdown element and work sequence are specified. On the other hand, a BPMN metamodel of activities and sequence flows have been specified in RSL for the definition of business processes. And finally, correspondence axioms have been defined, formally specified in RSL too, between the SPEM formal specification and the BPMN formal specification.

Our research is a contribution to the improvement of the management of the software development processes showing that the software development process is a particular business process. In our research, we use some standards. The standard SPEM used for the software development process specification. Standard BPMN used to model businesses processes. In this paper, we translate SPEM in BPMN. RSL is used for the formalization of this transformation. There are several transformations between BPMN and language languages executed by any workflow engine (as XPDL or BPEL). In this way, using our transformation between SPEM BPMN, the original software development process can be executed by a workflow engine. Therefore, the software development process is transformed into a workflow process specification that follows the standard of the WfMC, and standard workflow engines can be used to assist in the automatic management of the software development processes specified under the standard of the OMG, the SPEM.

This work improves the software development due to the automation by the workflow engine that will manage the resources and will organize a team of software engineers during the development of a particular project, allowing a better control of the work team activities. This automation is done for any software development process specified in SPEM.

10. References


