A Framework and Tool Support for Managing a Family of Business Process Variants

HENRY CHIKA ELEONU

Department of Computer Science, School of Engineering, The University of Manchester, Manchester, United Kingdom

Business organizations maintain business processes with multiple variants because of varied business requirements of which the support of these multiple business process variants constitutes a big challenge. BPPLE tool demonstration presents an extension of Eclipse BPMN modeller to cope with the modelling, and customization of business processes at build time and run time to compose the business process variants it may have. BPPLE tool is based on the Business Process Product Line Engineering (BPPLE) approach, our proposed approach for managing a family of business process variants. We have applied BPPLE in the scenarios such as the student registration in a higher education institution. Tests carried out showed that BPPLE tool enables the customization of business process models or instances to compose business process variant models or instances, respectively.

CCS CONCEPTS • Software and its engineering • Software creation and management • Software development techniques • Reusability • Software product lines

Keywords: Business process modelling, Business process modelling notation (BPMN), Business process management, Software engineering, Business rules, Software product line

1 INTRODUCTION

A business process is a set of activities performed in a specific sequence to realize a business goal and deliver value for a customer [29,30,54]. A business process model is a textual or graphical representation of a business process which enables the business process to be analyzed, improved and automated [23].

A fundamental challenge of business process modelling is to cope with multiple variants that may exist for a business process [43]. These multiple variants of a business process result from the need for business organizations to cope with changes in their environment, such as changes in regulations, laws, strategies, or operation [4]. These organizations need these changes to be reflected in the business process models by customizing them. However, there is a challenge in customizing business process instance at run time because of their inherent rigidity [55].

To capture changes in the business environment, organizations have traditionally maintained multiple variant models of a business process or expanded a business process model with condition branching (spit constructs) to accommodate the different scenarios of a business process [37]. The traditional ways of modelling business processes have limitations because it is challenging to have the best of numerous qualities in a business process model all at once. For instance, it is difficult to have minimized redundancy of activities, minimized complexity, and improved flexibility simultaneously [37]. Consolidating the multiple variant models results in models with higher complexity but minimized redundancy. Conversely, individually modelling the business process variants results in higher redundancy of activities but minimized complexity [37]. Popular business process management tools which follow the traditional way of modelling business processes have a limitation because they only enable the modelling of business process variants separately or modelling the business process variants by consolidating them in a single model [43].

A recent trend in the design of business process modelling frameworks is to find a compromise between consolidating business process variants or modelling the business process variants separately by using a reference model which can be customized to create a business process variant [26,59]. The limitation of approaches adopting reference model is that they are mostly theoretical and lacking implementation [45].

We propose a business process management approach called Business Process Product Line Engineering (BPPLE) which adopts the principles of Software Product Line (SPL) [41] and extends the conventional Business Process Management System (BPMS) architecture [55]. BPPLE decomposes a family of business process variants into a model of simple decoupled components of the generic reference model, a set of fragment models and business rules [27]. The generic reference model represents the commonalities for all business process variants in a family or the intersection of all business process variants in a family, and it has underspecified abstract containers. A business process variant can be composed by customizing the generic reference model by extending the abstract containers with the fragment models [27].

BPPLE approach aims to solve the problem of finding a balance between the complexity, redundancy and flexibility of the model of a family of business process variants, and also solve the problem of customization of business process instance at run time.

We present tool support for BPPLE approach for modelling a family of business process variants and customization of a generic reference model to compose business process variants at build time or run time.

2 SUPPORT TOOLS FOR BUSINESS PROCESS MODELLING APPROACHES

La Rosa et al. [45] wrote an extensive review of business process modelling approaches for handling variability of business processes with a description of their features which includes their support tools. We adapt the taxonomy of business process modelling approaches presented in [45] to group business process modelling approaches into configurable reference modelling, activity specialization, model segment customization.

Tool support for business process modelling approaches can support different level of customizations at run time such as restriction or extension [51]. Customization by restriction is the deleting of activities of a business process instance to restrict their behaviour and customization by extension is the adding of activities to a business process instance to extend their behaviour [51].

Support tools for business process modelling can either support planned or unplanned customization of the business process instance at run time [34]. For a tool to cope with unplanned customization, it must support the alternation between business process instance execution and business process modelling. The alternation between business process instance execution and business process modelling can allow a business process instance to be put in a wait state at run time so that changes can be made to parts of the business process that has not been reached by the execution so that when execution resumes, there are changes to the behaviour of the business process [34]. Unplanned customization of a business process instance at run-time involves the reengineering or evolution of the business process by adding of tasks which are unplanned at build time. For planned customization, the tasks needed to change the behaviour of a business process instance are planned at build time [35].

2.1 Configurable Reference Modelling

Approaches in this category use a single model with configurable nodes to represent the business process variants in a family where condition branching (split constructs) are used to model the alternative path of process execution. The single monolithic model has a limitation of encouraging complexity of a model. The nodes (function and split constructs) have annotations defined for them with semantics which can be changed. Examples of approaches in this category are the Configurable Event-Driven Process Chain (C-EPC) [47], the Configurative Process Modelling [5] and configurable workflow models [22].

Many approaches in this category are based on conceptual and non-executable business process modelling standard, for instance, C-EPC and Configurative Process Modelling are extensions of the Event-Driven Process Chain (EPC) modelling standard [40] which is conceptual and non-executable. Consequently, models based on them are hardly executed by the tools which support them; for instance, ARIS Business Architect [20] does not execute EPC models [46]. The tool support for approaches in this category can only possibly support run-time customization by restriction because the single reference model has already captured the envisaged process variants and left with the option of restricting parts of the model to compose a business process variant. Tools supporting approaches in this category have a limitation because they have no provision for the evolution of business process at run time [2,22,39].

2.2 Activity Specialization

Approaches in this category use a reference model to represent the commonalities between the variants in a family of business processes [42], where some of the activities in the reference model are partially specified and can be specialized based on the context. An arc connects the partially specified activity to a subprocess which specializes it [42]. Each of the subprocesses is tagged to indicate the feature it belongs to such that when a feature is enabled in the feature model, the corresponding subprocesses tagged with the feature are retained while other subprocesses with different feature tags are removed from the process model [42]. The reference model has modular parts (reference model and subprocesses) which make it different from the reference model of configurative reference modelling. Two approaches in this category are Process Family Engineering in Service-Oriented Applications (PESOA) [42] and Business Process Family Model (BPFM) [38].

Tool support for PESOA is described where the business process model is designed using a Java program [49]. The limitation of this tool support for PESOA is that it limits the ability of non-technical users to understand and manage the business process [49]. Tools for supporting approaches in this category lack support for the evolution of process instance because it is difficult for new subprocess models not planned at build time to be added to a reference process instance without terminating the process.

2.3 Model Segment Customization

Approaches in this category are based on the application of change operations on a base model to restrict or extend segments (parts) of a process model [26]. The restriction is achieved by deleting a segment of a process model, and extension is achieved by inserting a fragment into a process model. The base model is different from the reference model of approaches in the activity specialization category because it has no underspecified activities, and there is no arc connecting the fragments to the base model which means a decoupling of fragments from the base process model. One of the variant models of a family of business

processes can be adopted to be the base model [26]. Two approaches in this category are the Provop approach [26], and the process materialization approach [32].

A tool which is an extension of ARIS Business Architect is provided for the creation of Provop models. The tool enables knowledge workers to define change operations and group them in options which are applied to the base model to derive a customized model [43]. A limitation of approaches in this category is that they address customization of conceptual models which are not executable and they lack the mechanism for instance-level evolution of the business process since fragment must be defined at build time.

3 BPPLE APPROACH

Similar to Activity Specialization approaches, BPPLE uses an underspecified reference model, but in contrast, the fragments are decoupled from the reference model with no connecting arcs between them. Unlike Model Segment Customization approaches, the reference model of BPPLE is a generic reference model that is an intersection of all variants in a family of business process variants. We will describe the core assets of a BPPLE model, the BPPLE system architecture and the phases in developing a BPPLE model.

3.1 Core Assets of a BPPLE Model

To model a family of business process variants, they are separated into the core assets: the generic reference model, the set of fragments, the business rules, and the context information. The generic reference model, which is analogous to the common assets of SPL is the intersection of all variants of a family of business processes [25]. The generic reference model has abstract containers which are partially specified, and each abstract container is a generalization of variations in segments of the business process variants. A generic reference model can be reused to compose all the variants in a family of business processes.

A fragment model is a business process model designed to represent a segment of a business process variant with a variation, and it can specialize an abstract container in a generic reference model. Fragments are analogous to the components that are not common to all the products in SPL. There is also a unique fragment called a call fragment whose task is to bind other fragments to an abstract container at run time. A call fragment must have two essential activities, a business rule task which invokes the business rules and followed by a bind activity which invokes the fragment selected by the business rule task.

Business rules have the decision logic for selecting a fragment which will extend and abstract container, and they consist of a group of action enabler rules [24]. An action enabler rule evaluates a combination of context values in its condition part, and if they evaluate to a “true” value, then the action part is initiated, which includes obtaining a specified fragment. Each action enabler rule in a group evaluates a unique combination of context values.

Context information holds the data which characterizes the situation of a business process [13]. Context information consists of context variables, where a context variable is an object with a “key” and “value” properties where the “key” is the name of the context variable and the “value” hold the value of the context variable.

3.2 BPPLE System Framework

The BPPLE system architecture is illustrated in Figure 1 as an extension of the conventional business BPMS architecture [55], which is designed by adding the components: the business rules engine, the business rules repository and the business process variants generator.

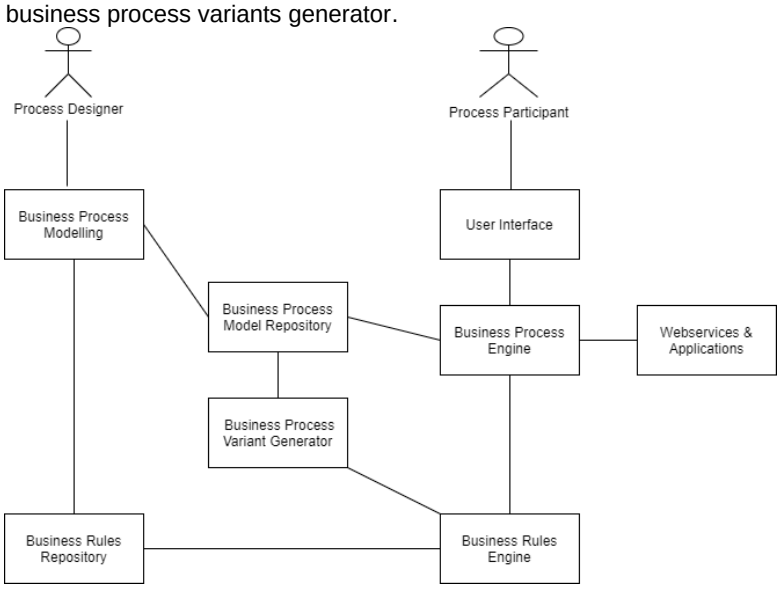

Figure 1: System Architecture for BPPLE

The business process modelling component is a business process definition tool that supports the process designer to model business processes, fragments or business rules. The business process and business process fragment models are stored in a business process model repository, and the business rules are stored in the business rules repository, see Figure 1.

The business rules engine executes the business rules held in the business rules repository to select a fragment. The business rules engine can receive requests from the business process engine or the business process variant generator to invoke the business rules. The business rules repository holds the business rules of a company.

The business process variant generator is the component which we developed, which uses the core assets for automatic composition of business process variant models at build time by inserting a fragment into each abstract container of a generic reference model. The variant generator communicates with the business rules engine to invoke business rules in the business rules repository.

The business process engine controls the execution of a business process instance. The business process engine first creates a business process instance from a business process model in the business process model repository, followed by the business process instance's execution. The business process engine can communicate with external applications, web services, and a client application's user interface. The business process engine communicates with the business rules engine to invoke business rules and communicates with the business process model repository to retrieve and invoke the required business process models or fragments.

The user interface enables process participant to communicate with a business process instance in the business process engine.

3.3 BPPLE Process Development Phases

BPPLE process development extends the life cycle of business process management put forward by Van Der Aalst et al. [3] by adding the variants composition phase after the system implementation phase, before the execution phase, and enabling the alternation between process modelling and process execution. Like SPL, the BPPLE process development occurs in two main stages: the core assets development and product development. Core assets development comprises two phases: process design and system implementation, and product development comprise three phases: variant composition, process execution, and process analysis.

3.3.1 Core Assets Development

Core asset development takes place at build time. In the process design phase, a business process's requirements are identified and documented in the form of a business process model to meet the business process's goal. The process design phase outputs are the generic reference model, the fragment models, and context information [3]. The system implementation phase focuses on business process automation where underlying systems are developed to support the generic reference model and fragment models. The system implementation phase outputs are the executable business process models, the executable fragment models, the scripts for the business rules, and the context information [3].

3.3.2 Product Development

Product development starts at build time with variant composition phase when the variant generator composes a business process variant model from the system implementation phase's outputs. If the composed business process variant is not completely composed, the composition will continue during the execution phase. Therefore, business process variant composition can take place at build time or run time, and there can be an overlap of the variant composition phase and the execution phase at run time.

(i) Variant Composition

The variant generator customizes the generic reference model in the variant composition phase by inserting a fragment into each abstract container to compose a business process variant. The variant generator invokes the business rules to select the business process fragments. Sometimes it not possible to customize an abstract container at build time because some required context values needed by the business rules to decide on a fragment are unknown, in which case a call fragment is inserted into affected abstract containers, and their customization is delayed to the execution phase.

(ii) Execution Phase

At the execution phase, the composed business process variant model, the output of the variant composition phase, is executed in the business process execution engine. The customization of a process instance can take place during process execution at run time when the business rules task of a call fragment in an abstract container invokes the business rules to select a fragment model, followed by the bind activity of the call fragment triggering the execution of the selected fragment model. At the execution phase, the values of the context variables whose values were unknown at build time are now known, making it possible for the business rules to select fragment models [1,3,15,31].

Figure 2 illustrates the run time architecture of BPPLE system showing business process instance execution, with the alternation between business process instance execution and business process modelling. A process participant starts the execution of a business process variant model through the Controller. If a pause activity is encountered during execution of a business process instance, the business process instance is put in a wait state to allow for the evolution of fragment models using the Business Process Modeller. When an input is received from the process participant using the user interface, the execution of the business process instance resumes, and when the business rules select the reengineered fragment models, they are invoked to change the behaviour of the business process instance.

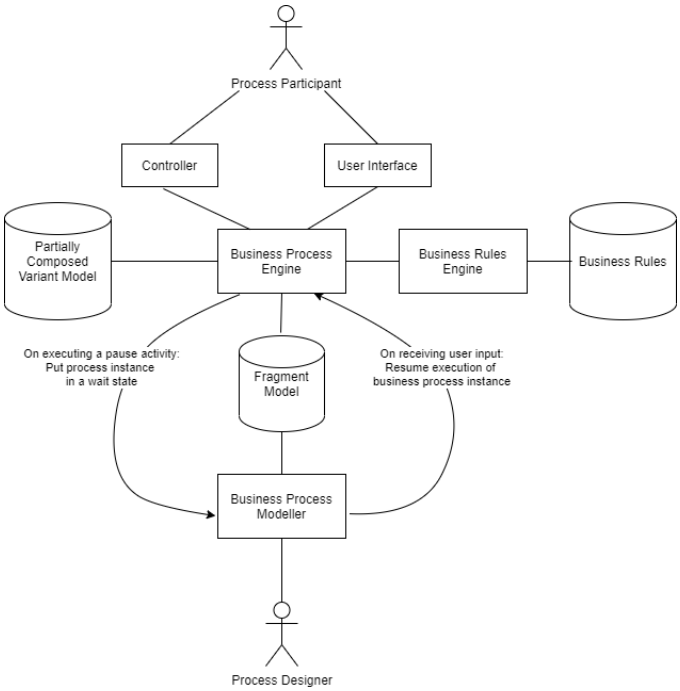

Figure 2: The run time architecture of BPPLE system showing business process instance execution

4 BPPLE TOOL

The BPPLE tool is presented with a running example of a student enrolment business process in higher education institutions to demonstrate its feasibility. The BPPLE tool demonstrates how an existing BPMS is extended with features to enable the customization of business process models and instances. We demonstrate how to model the core assets and customize the generic reference model to compose a business process variant.

To realize BPPLE tool, we used the Eclipse BPMN modeller as a basis and augmented it with features to enable customization of business processes at build time or run time. The Eclipse BPMN modeller is a plug-in of the Eclipse Integrated Development Environment (IDE) [18]. One advantage of using Eclipse IDE is

because it has an extensible plug-in system which enables the customization of the environment for different developments using several programming languages or standards [9,44].

The jBPM business process engine [33] is used to execute the business processes based on the BPMN2.0 standard [11,56]. The choice of BPMN as the language for modelling the generic reference model and fragments is because it is the most widely used standard by both business analyst and developers and it is a de facto standard for graphical modelling of business processes. BPMN models are represented graphically at a conceptual level, making it easy for non-technical user, and they also have the advantage of being executable [8,11,12].

We defined the business rule with the Drools Rule Language in a .drl text file executed in the Drools rule engine [48]. The choice of jBPM process engine and Drools rule engine is because they are compatible since the jBPM engine can easily send messages to or receive messages from the Drools rule engine to invoke business rules [48].

We augmented the Eclipse BPMN modeller with the variant generator and extended the BPMN language with new elements. The source code for the added features and implementation of some business processes are available on GitHub[1].

4.1 Implementation of Extensions for BPPLE System

We describe our implementation of the important extension to the conventional BPMS, which is the variant generator, and the extension of the BPMN elements with features that enable run time evolution of business process instance.

[1] https://github.com/henryeleonu/BPPLE_Projects

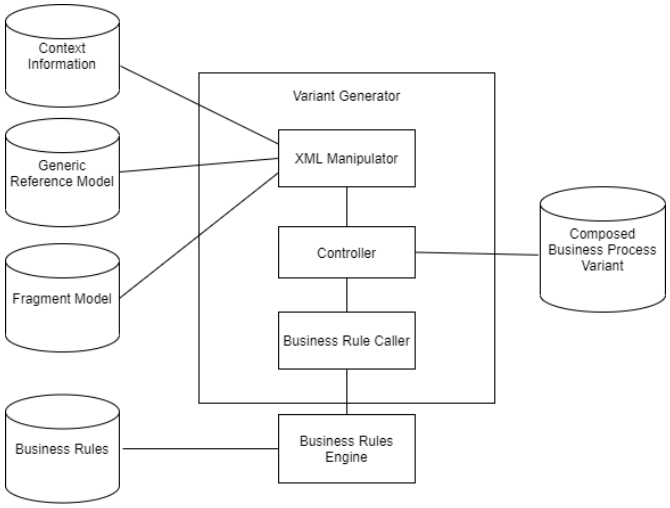

Figure 3: Architecture of the variant generator

4.1.1 Implementation of The Variant Generator

The variant generator is generic since it can be used to compose variants for different families of business processes. The variant generator is implemented with the Java programming language [21]. The structure of the variant generator program is represented with the architecture illustrated in Figure 3.

The Controller: coordinates the variant generation, which includes starting and terminating it. It invokes the XML Manipulator's functions and the Business Rules Caller and stores the composed business process variant in a repository.

The XML Manipulator: is used for manipulating the generic reference model, the fragments and the context information. It contains the operations for manipulating (cutting and merging) XML using XML Document Object Model (DOM) [17,53], and we use the Java API implementation of the XML DOM. The BPMN standard enables BPMN diagram models to be mapped to XML, so it is possible to manipulate the fragments and generic reference model with XML DOM [10].

Business Rules Caller: it invokes the business rules to be executed in the Business Rules Engine.

4.1.2 Extension of BPMN Elements

We created two new elements, the pause activity and the bind activity, which are extensions of the Task element of the BPMN 2.0 language. BPMN 2.0 introduces an extensibility mechanism that allows extending standard BPMN elements with additional attributes. We developed an underlining program called the work item handler for each of the new elements, which is the custom business logic for each of them [60]. The pause activity puts a business process instance in a wait state until input is received from a user through the user interface. The bind activity invokes the fragment model selected by the business rules task at run time.

4.2 Implementation of Core Assets

We describe the core assets' implementation with an example of student enrolment process in higher education institutions with several variants. The business process variants for student enrolment differ in the

candidate's financial support: self-funded candidates, candidates on scholarship and candidates on a government-funded student loan; candidates mode of entry: candidate transferring from another education institution, candidates who sat for examination test and candidates who did not take examination test; the educational institution which candidate applied to; and the course of study candidate applied to study. For instance, for some universities, candidates who applied for certain courses such as law or medicine must sit for admission test to be considered for admission. Figure 4 shows some variants for student enrolment into higher education institutions inspired by the candidate enrolment process in [50]. Figure 4a is the variant where candidates sit for admission test and financed by student loan. Figure 4b is the variant for a candidate that will not sit for admission test and are self-funded. Figure 4c is the variant where a candidate will not sit for admission test, is transferring from another higher education institution and is funded by a scholarship.

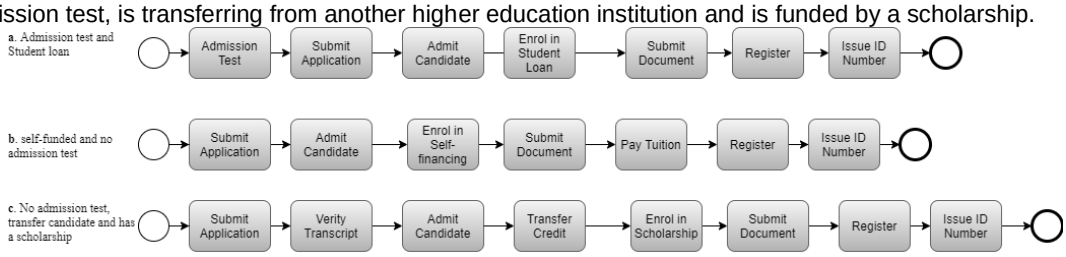

Figure 4: Process variants for student enrolment

The generic reference model and fragment models for the student enrolment process model is shown in Figure 5, where the grey activities are concrete, and in contrast, others are abstract containers. An abstract container of the generic reference model is modelled with the BPMN's sub-process element because it can serve as a container which holds a fragment [14]. The fragments are shown below their respective abstract containers of the generic reference model. The variant generator can insert, for instance, the "Admission Test" fragment into the "Admission Test" abstract container if admission test is needed for enrolment; otherwise, dumb activity (an activity with no function) is inserted into it.

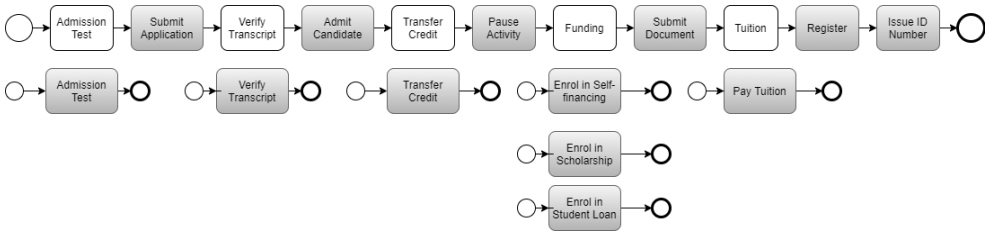

Figure 5: The generic reference model with fragments for the student enrolment process

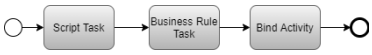

Figure 6: Screenshot of call fragment

Underlying programs are developed, embedded in the fragments and generic reference models' activities to support them. Variables are defined for the generic reference model such as the variable used to hold the

unique identity of fragment selected by the business rules for customizing an abstract container at run time. Figure 6 is a call fragment model where the first activity in the call fragment is the script task followed by the business rules task, and the last is the bind activity.

The context information is implemented in XML [57] with each <context> element holding the data of a context variable. The <context> element has a “key” attribute which holds the name of the context variable and a <value> element which holds the value of the context variable.

4.3 Composition of Business Process Variants

The business process variant model composed by the variant generator is shown in Figure 7 with the fragments inserted into the abstract containers.

5 EVALUATION

BPPLE tool was subjected to tests to determine whether it meets the quality requirements in terms of the executability and correctness of business process models composed by the variant generator and its ability to support the evolution of business process instances at run time. Three tests, executability test, syntactic correctness test and evolution test were carried out on three business processes, the student enrolment process, invoice verification process, and the construction plan process, see Table 1.

The executability of the composed business process variant models is tested with the assertProcessInstanceCompleted method of the jBPM testing framework, an extension of Junit test classes that enable testing of business processes defined in BPMN to ascertain whether the execution of a process instance is completed successfully [61]. An executability test is successful if the execution of a business process models is completed without any errors.

The business process models’ syntactic correctness is ascertained by validating them with the BPMN XML Schema Definition (XSD), a metamodel for validating the correctness of BPMN models [11,19]. A syntactic correctness test is successful if a business process model is validated without any errors or warnings.

TestNG testing framework [6,7] is used to test the evolution of a business process instance by performing a semantic test with the Assert.assertEquals method of TestNG after the user has designed a new fragment model at run time. The test is successful if the business process instance receives the expected message from the newly designed fragment model, showing that a fragment model that is unplanned at build time can be designed at run time to change the behaviour of a business process instance.

The three business processes passed the executability, syntactic, and evolution tests performed on them, see Table 1.

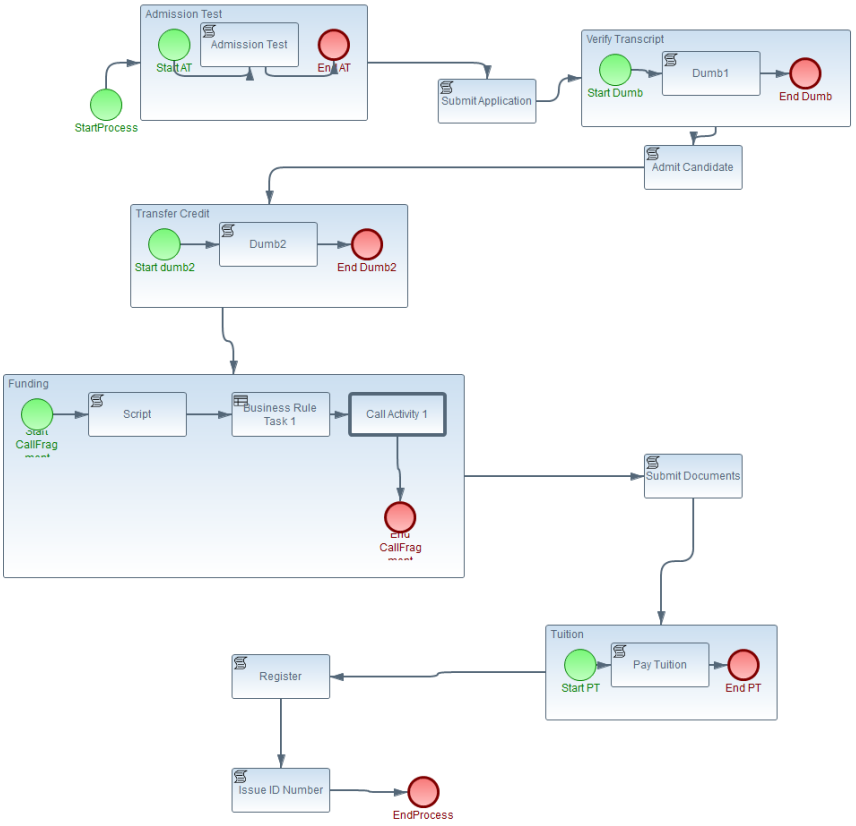

Figure 7: Student enrolment process variant composed by the variant generator

6 CONCLUSION

This paper presents the BPPLE framework and its proof-of-concept tool. The proof-of-concept tool allows users to create business process models, automatically generate business process variant models at build time and customize business process instance at run time. The findings from the results shown in Table 1 in section 5 show that the BPPLE tool can enable the composition of business process variant models that are syntactically correct and executable, and enable the evolution of a business process instance. From the findings, BPPLE framework will improve the management of a family of business process variant models by improving their maintainability.

In the future, we will explore how to scale the BPPLE tool to serve multiple clients in a Software as a Service model [58] and research the modelling of context information using ontology language [16,28,36,52].

Table 1: Test results for execution of the composed variant model, syntactic correctness of composed model and evolution of business process instance at run time

Business Process	Test for executability of a composed business process model with jBPM testing framework.	Test for syntactic correctness by validating the composed model with the BPMN XML schema.	Test for the evolution of a process instance with the TestNG testing framework after a new fragment model is designed at run time.
Student enrolment process	Test is successful	Test is successful	Test is successful
Invoice verification process	Test is successful	Test is successful	Test is successful
Construction plan process	Test is successful	Test is successful	Test is successful

Reference

[1] Wil M P Van der Aalst. 2013. Business process management: a comprehensive survey. *Int. Sch. Res. Not.* 2013, Article ID 507984 (2013), 37 pages. Retrieved from http://dx.doi.org/10.1155/2013/507984

[2] Wil M P Van Der Aalst and Arthur H M Ter Hofstede. 2005. YAWL: yet another workflow language. *Inf. Syst.* 30, 4 (2005), 245–275. Retrieved from https://doi.org/10.1016/j.is.2004.02.002

[3] Wil M P Van Der Aalst, Arthur H M Ter Hofstede, and Mathias Weske. 2003. Business process management: A survey. In *Business Process Management*, Springer Berlin Heidelberg, Berlin, Heidelberg, 1–12. Retrieved from https://doi-org.manchester.idm.oclc.org/10.1007/3-540-44895-0_1

[4] G.H. Alférez, V. Pelechano, R. Mazo, C. Salinesi, and D. Diaz. 2014. Dynamic adaptation of service compositions with variability models. *J. Syst. Softw.* 91, (May 2014), 24–47. DOI:https://doi.org/10.1016/j.jss.2013.06.034

[5] Jörg Becker, Patrick Delfmann, Alexander Dreiling, Ralf Knackstedt, and Dominik Kuropka. 2004. Configurative process modeling–outlining an approach to increased business process model usability. In *Proceedings of the 15th IRMA International Conference*, Gabler New Orleans, 1–12.

[6] Cédric Beust and Hani Suleiman. 2007. *Next generation java testing: TestNG and advanced concepts*. Pearson Education.

[7] Purnima Bindal and Sonika Gupta. 2014. Test Automation Selenium WebDriver using TestNG. *J. Eng. Comput. Appl. Sci.* 3, 9 (2014), 18–40.

[8] Paolo Bocciarelli and Andrea D'Ambrogio. 2011. A BPMN extension for modeling non functional properties of business processes. In *Proceedings of the 2011 Symposium on Theory of Modeling & Simulation: DEVS Integrative M&S Symposium*, Society for Computer Simulation International, 160–168.

[9] Ed Burnette. 2005. *Eclipse IDE Pocket Guide: Using the Full-Featured IDE* (revised ed.). " O'Reilly Media, Inc." Retrieved from https://books.google.co.uk/books?id=MUQyvAIn5YQC

[10] Michele Chinosi and Alberto Trombetta. 2009. Modeling and validating BPMN diagrams. In *2009 IEEE Conference on Commerce and Enterprise Computing*, IEEE, 353–360. DOI:https://doi.org/10.1109/CEC.2009.48

[11] Michele Chinosi and Alberto Trombetta. 2012. BPMN: An introduction to the standard. *Comput. Stand. Interfaces* 34, 1 (2012), 124–134. Retrieved from https://doi.org/10.1016/j.csi.2011.06.002

[12] Gero Decker, Remco Dijkman, Marlon Dumas, and Luciano García-Bañuelos. 2008. Transforming BPMN diagrams into YAWL

nets. In *International Conference on Business Process Management*, Springer, 386–389.

[13] Anind K Dey, Gregory D Abowd, and Daniel Salber. 2001. A conceptual framework and a toolkit for supporting the rapid prototyping of context-aware applications. *Human–Computer Interact.* 16, 2–4 (2001), 97–166. DOI:https://doi.org/10.1207/S15327051HCI16234_02

[14] Remco M Dijkman, Marlon Dumas, and Chun Ouyang. 2008. Semantics and analysis of business process models in BPMN. *Inf. Softw. Technol.* 50, 12 (2008), 1281–1294. DOI:https://doi.org/https://doi.org/10.1016/j.infsof.2008.02.006

[15] Marlon Dumas, Marcello La Rosa, Jan Mendling, and Hajo A Reijers. 2018. *Fundamentals of business process management* (Second Edi ed.). Springer. Retrieved from https://doi.org/10.1007/978-3-662-56509-4

[16] Henry Eleonu and Jane Oruh. 2013. Adaptive Process in a Pervasive System-A Holistic Hybrid Approach. *Int. J. Comput. Sci. Issues* 10, 3 (2013), 208.

[17] Jeff Friesen. 2019. Parsing and Creating XML Documents with DOM. In *Java XML and JSON: Document Processing for Java SE*. Apress, 67–112. DOI:https://doi.org/10.1007/978-1-4842-4330-5_3

[18] David Geer. 2005. Eclipse becomes the dominant Java IDE. *Computer (Long. Beach. Calif).* 38, 7 (2005), 16–18. DOI:https://doi.org/10.1109/MC.2005.228

[19] Matthias Geiger and Guido Wirtz. 2013. Detecting Interoperability and Correctness Issues in BPMN 2.0 Process Models. In *ZEUS*, 41–44.

[20] Saiedeh Ghatrei. 2015. ARIS Enterprise Architecture's Usage Reviews. *Lect. Notes Softw. Eng.* 3, 1 (2015), 57. DOI:https://doi.org/10.7763/LNSE.2015.V3.166

[21] James Gosling, Bill Joy, Guy Steele, and Gilad Bracha. 2000. *The Java language specification*. Addison-Wesley Reading.

[22] Florian Gottschalk, Wil M P Van Der Aalst, Monique H Jansen-Vullers, and Marcello La Rosa. 2008. Configurable workflow models. *Int. J. Coop. Inf. Syst.* 17, 02 (2008), 177–221.

[23] Volker Gruhn and Ralf Laue. 2006. Complexity metrics for business process models. In *Lecture Notes in Informatics (LNI), Proceedings - Series of the Gesellschaft fur Informatik (GI)*, Citeseer, 1–12.

[24] Barbara Von Halle. 2001. *Business rules applied: building better systems using the business rules approach* (1st ed.). Wiley Publishing.

[25] Alena Hallerbach, Thomas Bauer, and Manfred Reichert. 2009. Issues in modeling process variants with provop. In *Business Process Management Workshops*, Springer Berlin Heidelberg, Berlin, Heidelberg, 56–67. DOI:https://doi.org/10.1007/978-3-642-00328-8_6

[26] Alena Hallerbach, Thomas Bauer, and Manfred Reichert. 2010. Capturing variability in business process models: the Provop approach. *J. Softw. Maint. Evol. Res. Pract.* 22, 6-7 (2010), 519–546.

[27] David Hay and Keri Anderson Healy. 2000. *Defining business rules - what are they really*. Business Rule Group. Retrieved from https://www.businessrulesgroup.org/first_paper/BRG-whatisBR_3ed.pdf

[28] Ian Horrocks, Peter F Patel-Schneider, and Frank Van Harmelen. 2003. From SHIQ and RDF to OWL: The making of a web ontology language. *J. Web Semant.* 1, 1 (2003), 7–26. DOI:https://doi.org/https://doi.org/10.1016/j.websem.2003.07.001

[29] Chan Meng Khoong and William A. Wheeler. 1997. Business process reengineering: Breakpoint strategies for market dominance. *Interfaces (Providence).* 27, 3 (1997), 112–114. Retrieved from http://www.jstor.org/stable/25062263

[30] Mathias Kirchmer. 2017. *High performance through business process management* (3rd ed.). Springer International Publishing. DOI:https://doi.org/10.1007/978-3-319-51259-4

[31] Ryan K L Ko, Stephen S G Lee, and Eng Wah Lee. 2009. Business process management (BPM) standards: a survey. *Bus. Process Manag. J.* 15, 5 (2009), 744–791. DOI:https://doi.org/10.1108/14637150910987937

[32] Akhil Kumar and Wen Yao. 2009. Process materialization using templates and rules to design flexible process models. In *Rule*

Interchange and Applications, Springer Berlin Heidelberg, Berlin, Heidelberg, 122–136. DOI:https://doi.org/10.1007/978-3-642-04985-9_13

[33] Mariano Nicolas De Maio, Mauricio Salatino, and Esteban Aliverti. 2014. *jBPM6 Developer Guide*. Packt Publishing Ltd.

[34] Andrea Marrella. 2018. What automated planning can do for business process management. In *Business Process Management Workshops*, Springer International Publishing, 7–19. DOI:https://doi.org/10.1007/978-3-319-74030-0_1

[35] Andrea Marrella, Massimo Mecella, and Sebastian Sardina. 2016. Intelligent process adaptation in the SmartPM system. *ACM Trans. Intell. Syst. Technol.* 8, 2 (2016), 1–43. DOI:https://doi.org/10.1145/2948071

[36] D L Mcguinness, R Fikes, J Hendler, and L A Stein. 2002. DAML+OIL: an ontology language for the Semantic Web. *IEEE Intell. Syst.* 17, 5 (2002), 72–80. DOI:https://doi.org/10.1109/MIS.2002.1039835

[37] Fredrik Milani, Marlon Dumas, Naved Ahmed, and Raimundas Matulevičius. 2016. Modelling families of business process variants: a decomposition driven method. *Inf. Syst.* 56, (2016), 55–72. DOI:https://doi.org/https://doi.org/10.1016/j.is.2015.09.003

[38] Mikyeong Moon, Minwoo Hong, and Keunhyuk Yeom. 2008. Two-level variability analysis for business process with reusability and extensibility. In *2008 32nd Annual IEEE International Computer Software and Applications Conference*, IEEE, 263–270. DOI:https://doi.org/10.1109/COMPSAC.2008.129

[39] Vanessa Tavares Nunes, Flávia Maria Santoro, Claudia Maria Lima Werner, and Célia Ghedini Ralha. 2016. Context and planning for dynamic adaptation in PAIS. In *Business Process Management Workshops*, Springer International Publishing, Cham, 471–483. DOI:https://doi.org/10.1007/978-3-319-42887-1_38

[40] Markus Nüttgens, Thomas Feld, and Volker Zimmermann. 1998. Business Process Modeling with EPC and UML: transformation or integration? In *The Unified Modeling Language*, Martin Schader and Axel Korthaus (eds.). Physica-Verlag HD, 250–261. DOI:https://doi.org/10.1007/978-3-642-48673-9_17

[41] Klaus Pohl, Günter Böckle, and Frank J van Der Linden. 2005. *Software product line engineering: foundations, principles and techniques*. Springer Science & Business Media.

[42] Frank Puhlmann, Arnd Schnieders, Jens Weiland, and Mathias Weske. 2005. *Variability mechanisms for process models*.

[43] Manfred Reichert, Steve Rechtenbach, Alena Hallerbach, and Thomas Bauer. 2009. Extending a business process modeling tool with process configuration facilities: The Provop demonstrator. In *CEUR Workshop Proceedings*.

[44] Jim des Rivières and John Wiegand. 2004. Eclipse: A platform for integrating development tools. *IBM Syst. J.* 43, 2 (2004), 371–383. DOI:https://doi.org/10.1147/sj.432.0371

[45] Marcello La Rosa, Wil M P Van Der Aalst, Marlon Dumas, and Fredrik P Milani. 2017. Business process variability modeling: A survey. *ACM Comput. Surv.* 50, 1 (2017), 2. DOI:https://doi.org/10.1145/3041957

[46] Marcello La Rosa, Marlon Dumas, Arthur H M Ter Hofstede, and Jan Mendling. 2011. Configurable multi-perspective business process models. *Inf. Syst.* 36, 2 (2011), 313–340. Retrieved from https://doi.org/10.1016/j.is.2010.07.001

[47] Michael Rosemann and Wil M P van der Aalst. 2007. A configurable reference modelling language. *Inf. Syst.* 32, 1 (2007), 1–23. DOI:https://doi.org/https://doi.org/10.1016/j.is.2005.05.003

[48] Mauricio Salatino, Mariano De Maio, and Esteban Aliverti. 2016. *Mastering jboss drools 6*. Packt Publishing Ltd.

[49] Arnd Schnieders and Frank Puhlmann. 2006. Variability mechanisms in e-business process families. In *Business Information Systems–9th International Conference on Business Information Systems (BIS 2006)*, Gesellschaft für Informatik eV, Bonn, 583–601.

[50] Nataša Subic and Maja Dimitrijevic. 2015. Modeling flexible configurable processes applied to the enrollment process in higher education institutions. *Online J. Appl. Knowl. Manag.* 3, 2 (2015), 181–190. Retrieved from http://www.iiakm.org/ojakm/articles/2015/volume3_2/OJAKM_Volume3_2pp181-190.pdf

[51] Mikael Svahnberg and Jan Bosch. 2000. Issues concerning variability in software product lines. In *Software Architectures for Product Families*, Springer Berlin Heidelberg, Berlin, Heidelberg, 146–157. DOI:https://doi.org/10.1007/978-3-540-44542-5_17

[52] Jorge E López De Vergara, Víctor A Villagrá, and Julio Berrocal. 2004. Applying the Web Ontology Language to management information definitions. *IEEE Commun. Mag.* 42, 7 (2004), 68–74. DOI:https://doi.org/10.1109/MCOM.2004.1316535

[53] Fangju Wang, Jing Li, and Hooman Homayounfar. 2007. A space efficient XML DOM parser. *Data Knowl. Eng.* 60, 1 (2007), 185–207. Retrieved from https://doi.org/10.1016/j.datak.2006.01.002

[54] M Weske. 2012. *Business process management: Concepts, languages, architectures, second edition* (3rd ed.). Springer. Retrieved from https://doi.org/10.1007/978-3-662-59432-2

[55] Mathias Weske. 2012. Business process management architectures. In *Business Process Management*. Springer, Berlin, Heidelberg, 333–371. DOI:https://doi.org/10.1007/978-3-642-28616-2_7

[56] Stephen A White. 2004. Introduction to BPMN. *Ibm Coop.* 2, 0 (2004), 0.

[57] François Yergeau, Tim Bray, Jean Paoli, C Michael Sperberg-McQueen, and Eve Maler. 2004. Extensible markup language (XML) 1.0. *W3C Recomm.* 4, (2004), 220.

[58] Dongjin Yu, Qi Zhu, Dalong Guo, Binbin Huang, and Jianwen Su. 2015. jBPM4S: A multi-tenant extension of jBPM to support BPaaS. *Asia Pacific Bus. Process Manag. AP-BPM 2015. Lect. Notes Bus. Inf. Process.* 219, (2015), 43–56. DOI:https://doi.org/10.1007/978-3-319-19509-4_4

[59] Jian Yu, Quan Z Sheng, Joshua K Y Swee, Jun Han, Chengfei Liu, and Talal H Noor. 2015. Model-driven development of adaptive web service processes with aspects and rules. *J. Comput. Syst. Sci.* 81, 3 (2015), 533–552. Retrieved from https://doi.org/10.1016/j.jcss.2014.11.008

[60] Chapter 21. Domain-specific Processes. Retrieved August 19, 2020 from https://docs.jboss.org/jbpm/v6.0/userguide/jBPMDomainSpecificProcesses.html

[61] Chapter 14. Testing and debugging. Retrieved May 30, 2020 from https://docs.jboss.org/jbpm/v5.2/userguide/ch14.html