Context-Aware Process Modelling through Imperative and Declarative Approach

Olaolu Sofela, Lai Xu, and Paul De Vrieze

Software Systems Research Centre, Bournemouth University, UK
{osofela,lxu,pdvrieze}@bournemouth.ac.uk

Abstract. Business process models can help to improve process visibility and shared understanding, particularly across internal and external stakeholders. By supporting knowledge transfer and enhancing documentation, they aid compliance and ultimately improve software quality. However, many current process modelling approaches and tools require significant learning on the part of users and resultant models may often be prone to errors. The use of such approaches also demands a certain level of domain expertise in both the business and IT domains. Hence, the need to accommodate a wider user group for business process models, including those, often end users, who are not IT experts. Context-aware process modelling allows end users to identify a business process model from a process repository according to different context information. In this paper, we develop a new approach to help end users for building up potential process models using PVDI (process variability through declarative and imperative). A real world case is used to explain and verify our approach.

Keywords: Collaborative process modelling, Declarative process modelling, Context-aware process modelling, Virtual business process modelling, and Virtual collaborative process modelling.

1 Introduction

As many organisations have reached higher levels of business process management maturity, they tend to collect and actively use large numbers of business process models [1]. Enterprise software vendors and business process management consultancies have also built a larger number of business process reference models for requirements analysis, communication, automation, compliance, etc. Such collections of industry-strength business process models and collaborative business process models include thousands of activities and related business objects such as data and business artefacts.

The questions such as, how to manage such large collections of process models and how to reuse them to facilitate the development of new collaborative process models, provides new challenges and opportunities to the area of business process management. Context aware process modelling is one approach to store different process models in a process repository, to represent a process models, and to identify

L.M. Camarinha-Matos and R.J. Scherer (Eds.): PRO-VE 2013, IFIP AICT 408, pp. 191–200, 2013.

process models from a process repository only once while specifying the differences depending on specific context categories (such as business domain, business industry, country, business role and so on.)

Context awareness is an essential role for the dynamic appearance of structures and the dynamic linguistic representation of components. In this paper, we focus on providing dynamic collaborative business process structures according to the context information. Collaborative process structures embrace context awareness aspects of both the context driven flow of collaborative business processes and contained activities. In addition, relations among activities and other business artefacts are context dependent.

We introduce a new approach to facilitate end users to build their collaborative business process models. From our experience in building collaborative business process models, there are always alternative collaboration process models. How to completely present all possible models and how to select potential suitable collaboration process models are the research problems we try to solve in this paper. We have used both imperative and declarative process modelling approaches to solve the problem.

1.1 Related Work

The context awareness business process model is used to achieve reusability and consistency. Some initial approaches of the consideration of contexts within business process models can be found within [2, 3, 4]. Saidani et. al. present how to perform business process modelling using context aware information during all modelling stages [4]. The steps of supporting the context aware process modelling include context elicitation, context categorisation, context adaptation and measure, and business process instantiation. In [5] the authors concentrate on the context-relevant adjustment of configuration variants of the technical execution of business processes; whereas in [6] the authors deal with the situations which affect the flow of business process models. These are initial efforts on context-aware business process modelling. They did not consider structural differences and how context methods can actually change the flow of these models. Further, these aspects primarily concentrate on business processes within companies and they primarily focus on their context-driven flow.

The context awareness aspects which are developed within our work consider the context-driven differences in the structural appearance of business process models and focus on various levels as well as context-driven terminologies for business artefacts. Furthermore, our proposal does not focus on any specific application of contexts and neither on any specific context categorisation; rather it contemplates context-driven differences in general on an abstract level.

Paper [7] and [8] present a methodology for automatically generating business processes which bring a declarative perspective to introduce a variability framework for BPM. The authors use temporal logic to formalise partial orders among activities of a process. The choice and customization is open for the adaption. We adopt PVDI (process variability through declarative and imperative) from [7, 8] to general variable collaborative process models in our context-aware process modelling.

The rest of the paper is organised as follows: Section 2 provides a motivational case. The proposed imperative and declarative approach is explained in Section 3. An evaluation of our approach is then described in Section 4 by using our motivational case. Last, the conclusion and the future work are drawn.

2 Case Study

In United Kingdom, there are 7,500 companies in the plastics industry, each of these companies produces this plastic by different ways, e.g. extrusion, thermoset processing, injection moulding and few other more, but the way they run their day transaction is same.

Figure 1 illustrates a simplified version of the general and visible way of any plastic company process. The simplified version shows the collaboration between various companies. A company requests a tonne of plastic bag to be made Plastics Ltd, the company receives the purchase order (A), checks the availability of the needed raw materials (Polyethylene, recycled products) (A2), also check account history of the customer.

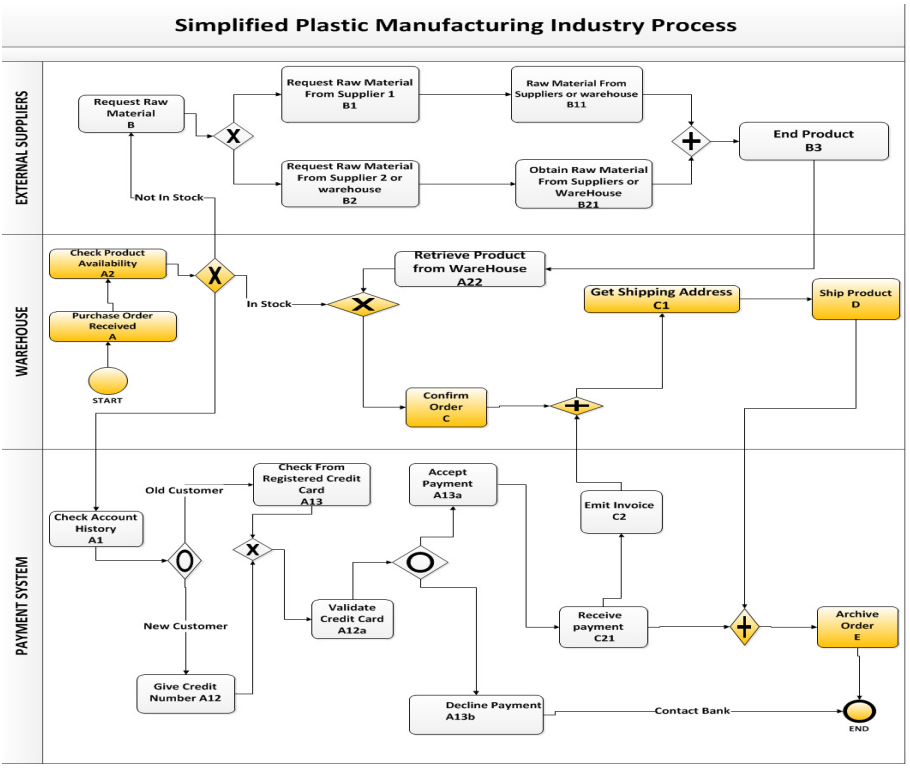

Fig. 1. PMI Process Model

The stock level determines whether Plastic Ltd requests more raw materials from their suppliers (B1), (B2) to manufacture more of the plastic bag. If the stock level is enough, then the tonne of plastic bag is retrieved from the warehouse (A22 and the order is confirmed (C). Next, all ready to be shipped to the given address (C1). Before the plastic bags can be shipped off, the payment is needed to be confirmed by the account systems.

Within the account systems, a record of companies and credit card (A13), are kept in a secured database, which different a new customer and old customer, a new company has to give its company details, also its credit card number (A12), which will be automatically validated (A12a), for every transaction. An accepted payment A13a will precede the confirmed order to be shipped off (D), but a declined payment will automatically end the process and the company will be notified.

The case provides changeable business processes according different business situations. In the following sections, we show how the collaboration process can be configured.

3 Configuring Imperative Process Models

Imperative process modelling takes an `inside-to-outside' approach: all execution alternatives are explicitly specified in the model and new alternatives must be explicitly added to the model [11]. Imperative process modelling provides sequential information what the process flow should go. In an imperative process model the model explicitly specifies all valid execution paths. On the other hand, *declarative process modelling* takes an `outside-to-inside' approach: constraints implicitly specify execution alternatives as all alternatives that satisfy the constraints and adding new constraints usually means discarding some execution alternatives [11]. Declarative process modelling provides the rules what the process should not do. In a declarative process model the system is constrained by conditions for valid code paths. There could be an infinite amount of valid paths and enumerating valid paths may be impossible.

In this section, we introduce our approach of how possible process models are built according to different guides in different contexts. The three steps are followed.

1. Based on an as-is process model and the case study to abstract logic and temporal dependencies of the imperative process model (see Section 3.1).
2. Identify frozen group which are core parts of business and identify close area which are possible to change and flexible (see Section 3.2).
3. Use PVDI (process variability through declarative and imperative) to verify all potential collaborative process models (see Section 3.3).

3.1 Logical and Temporal Dependencies

Logical and temporal dependencies are defined as a set of rules which creates a component that depends on a specific set of pre-conditions. It is mandatory sometimes to remove the activity when those pre-conditions no longer hold. Logical

dependencies help to prevent errors in configuration [9], which the requirements, dependency and prohibitions rules must be met.

Definition 1 (Requirements). Let A and B are activities, Requires: A →B represents that if activity B is chosen during configuration, then activity A must be chosen.

Definition 2 (Dependency). Let A and B are activities, Depends: A→ B means that if activity B is not chosen during the configuration, activity A must not be chosen.

Definition 3 (Prohibition). Let A and B are activities, Prohibits: A →B that if activity A is chosen during configuration, activity B must not be chosen, vice-versa.

Temporal dependencies define whether a particular activity has to be performed before or after another activity, depending on the exact service module to be executed at a given time [10].

Definition 4 (Precedence). Let A and B are activities, Before (A)→ B denotes that if activity B must be occurred before activity A occurs.

Definition 5 (Direct Precedence). Let A and B are activities, iBefore (A) → B signifies that activity B must be occurred directly before activity A.

Definition 6 (Succession). Let A and B are activities, After (A)→ B acts for execute activating B after activity A

Definition 7 (Direct Succession). Let A and B are activities, iAfter (A) → B describes that activity B executes immediately after activity A.

Further examples of logic dependencies and temporal dependencies of the PMI process are provided in Section 4.1.

3.2 Frozen Group and Closed Area

Constraints are set of valid valuations drawn from a process, which must be TRUE. The constraints are needed to be known, in further understanding of this approach, frozen group and closed area are need to be known, from this constraints the variability of a process can be built.

A *frozen group* is a specific area which the given designed template may not be modified. Every node from the start and end paths in a frozen area are mandatory to be selected in the template [8] and is frozen which is not subject to changes.

For example, the frozen group of the PMI process is the darken flow in Figure 1, which are the primary activities details of the PMI company which cannot be altered, Figure 2.

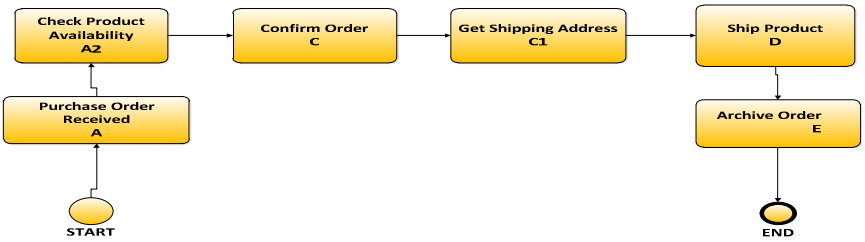

Fig. 2. Frozen Area of the PMI process

A *closed area* (CA) is imperative specification that provides possible modifications that is, it may allow changes to the elements of the business process. It is a selected area constrained between the incoming and exactly one outgoing flow to and from it, in which every nodes are mandatory between to select [8]. The closed areas are normally classified by different functions of different internal or external partners.

For example, the set of constraints developed over set activities within PMI group, related to the 3 parts, $PMI_{payment}$, $PMI_{warehouse}$, and $PMI_{exteranl\_supplier}$ are shown below

a) $PMI_{payment} = \{ A, A_{12}, A_{13}, A_{12a}, A_{13a}, C_2, C_{21}, E \}$
b) $PMI_{warehouse} = \{ A, A_{22}, C, C_1, D, F \}$
c) $PMI_{external\_supplier} = \{B, B_1, B_2, B_{11}, B_{21}, B_3\}$

3.3 Different Process Variability Approaches

Variability is abstraction in which organization preserve its standard business processes but also allows other templates to be built, customized and adapted on the exciting processes. The goal of variability is to support processes to be re-usable and flexible; the current three variability approaches are namely: imperative variability, declarative variability, and Process Variability through Declarative and Imperative Technique (PVDI).

Imperative variability is an approach in which variables may or may not be added to an existing template at design time. The imperative view on variability within the PMI process is shown in Figure 3a and 3b which explains the variability within the external supplier and the payment system respectively, the warehouse section has no variability that is it cannot be modified.

(a) Simplified Possible Model 1 (b) Simplified Possible Model 2

Fig. 3. Two Simplified Possible PMI Process Variants

In Figure 3a, activity (B) notes 'Request Raw Material' activity, (X) describes the 'Forward Sub-Order' sub-processes from different warehouses and S represents the 'Request Raw Material' sub-processes from different suppliers. The new possible variants (X_1), (X_2) for sub-process (X), (S_1), (S_2) for sub-process S may or may not be added to the initial template depending on the reference process and context information [9].

In Figure 3b, the variants in the payments systems are as follows: (A_{12b}) represent the 'Cheque Payment' activity; (A_{12c}) describes 'Invoice Number'; A_{12d} represents 'Validate Cheque or Invoice Number'; and (C_{22}) represents 'Refund'. Each activity or sub-processes can be replaced according to the process execution. Using imperative modelling approach, such variants need to be pre-defined and be configured according to certain mechanism.

Declarative variability is a brand new way of designing business processes where the basic principles or requirements of a process in a template are defined; variants are created and validated from the template.

PVDI is a relative new technique that allows a high degree of process variability while preserving the main business goal of a process [8], while preserving the designed templates. Groefsema et al [7, 8] provided a new approach to making a business process more flexible, it combines the properties of imperative, declarative variability and newly developed process modelling environment with graphical elements.

When end users want to build a process model for their situational needs, they provide context information which a mate-level process model can be identified. They can select the main parts or frozen parts of business activities or sub-processes and identify the flexible parts or close area. Logic and temporal dependencies can be created according to guides. Based on the frozen group, close areas, and logic temporal dependencies, the activity floating will be generated according to our logic and temporal dependencies. Until all constraints (frozen group, close areas, and logic and temporal dependencies) are satisfied.

4 Evaluation

In Section 3, we introduce the principle of how to configure the variable process models. In this section, we provide logical temporal dependencies from the PMI case and one example of variable sub-processes because the page limitations.

4.1 Logical and Temporal Dependencies for the Case

In executing the processes in Figure 2, there are sets of rules that must be known, it is necessary to note that the logical and temporal dependencies are formalised during the configuration which is shown in the Table 1.

Table 1. Logical and temporal dependencies for the PMI case

Logical Dependencies		
RULE	FORMALISATION	EXPLANATION
Requirements (Before)	Requires:B11 → B1 Requires B21 → B2 Requires:C2 → C21	Activity B1 (‘Request Raw Material from Supplier 1’) is only *required* after Activity B (‘Requires Raw Material’)
Dependency	Depends: A12a → A12 Depends: C21 → A22 Depends: C2 → C21	From payment system, Activity A12a ('Validate Credit Card') *depends* on Activity A12 ('Give Credit Card ')
Prohibition	Prohibits: A13a → A13b	From the payment System, Activity A13a ('Accept Payment') *prohibits* Activity A13b('Decline Payment') or vice versa.

Table 1. *(Continued)*

Temporal Dependencies		
RULE	FORMALISATION	EXPLANATION
Precedence	Before (C or B) →A1 *(This rule must conform to the Country's Business Law, might be vice-versa for certain countries*). i.e. A1 →(C or B)	Activity A1 ('Check Account History in Payment System') must done must be executed *first before* Activity C or B ('Confirm Order in Warehouse system or Request Raw Material in the Supplier System respectively').
Direct Precedence	iBefore (C) →A22 iBefore (A12a) → A13 or A12	In the Warehouse System, Activity A22 ('Retrieve Product') is executed *directly before* Activity C ('Confirm Order')
Succession	After (A) →A2 After (C21) →C2 After C2→C1 or D	In the Warehouse System, Activity A2 ('Check Product Availability') is executed *after* Activity A ('Purchase Order Received').
Direct Succession	iAfter (A12a) →(A13a v A12)	In payment System, Activity A12a ('Validate Credit Card') is executed *directly after* activity A13a ('Accept Payment') or Activity A12 ('Give Credit Number')

4.2 Process Variability

Within the PMI process as shown in Figure 1, two variants were discovered through cross examinations of the possibility of flexibility within the PMI process. We split Figure 1 into two parts, namely *suppliers section* and the *payment section* which all of these parts are tested, ignoring the warehouse section because it is a closed area, i.e. every node in this section is mandatory to be selected [8]. The suppliers and payment sections called variant 1 and variants 2 respectively maybe or may not be implemented into the simplified PMI process in Figure 1, depending on the business needs.

The external supplier sections is can be said to be flexible and easy to re-use i.e. if there are needs to request a raw material from more than two suppliers or other warehouses own by the company, the diagram is re-modelled to show the possible variants 1 with new configurations shown in Figure 4.

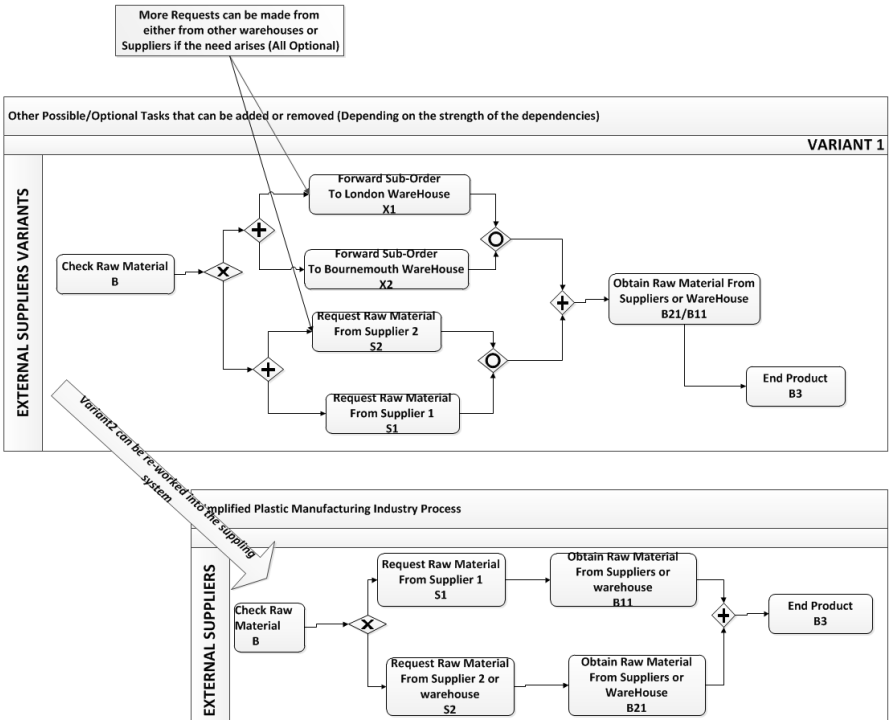

Fig. 4. Imperative Views on Variability within the External Supplier

5 Conclusion

Context-aware process modelling has been conceptually discussed and partially implemented in our previous work [12, 13]. This paper has presented a new approach to guide end users for building their collaborative business process models or collaborative process models using both declarative and imperative modelling methods. The aim of the approach is to provide process visibility for different business situations. We are fully exploiting the context aware process modelling method which not only provides process model visibility, but combines with workflow pattern based model building approach to build a solid process model for the end users.

Acknowledgements. This work is made possible by the support of the Natural Science Foundation of China (NSFC) under Grant No.61150110484, ESSENTIAL: Enterprise Service deSign based on ExistiNg software Architectural knowLedge.

References

1. Process Model Collections (2013), `http://processcollections.org/`
2. Rosemann, M., Recker, J.C., Flender, C., Ansell, P.D.: Understanding context-awareness in business process design (2006)

3. Rosemann, M., Recker, J.C.: Context-aware process design: Exploring the extrinsic drivers for process flexibility. In: The 18th International Conference on Advanced Information Systems Engineering. Proceedings of Workshops and Doctoral Consortium. Namur University Press (2006)
4. Saidani, O., Nurcan, S.: Towards context aware business process modelling. In: 8th Workshop on Business Process Modeling, Development, and Support (BPMDS 2007), Electronic Resource (2007) (last accessed October 22, 2007)
5. Hallerbach, A., Bauer, T., Reichert, M.: Context-based configuration of process variants, pp. 31–40 (2008)
6. Rosemann, M., Recker, J., Flender, C.: Contextualisation of business processes. International Journal of Business Process Integration and Management 3(1), 47–60 (2008)
7. Groefsema, H., Bulanov, P., Aiello, M.: Declarative Enhancement Framework for Business Processes. In: Kappel, G., Maamar, Z., Motahari-Nezhad, H.R. (eds.) ICSOC 2011.LNCS, vol. 7084, pp. 495–504. Springer, Heidelberg (2011)
8. Groefsema, H., Bulanov, P., Aiello, M.: Imperative versus Declarative Process Variability: Why Choose? (2012)
9. Aiello, M., Bulanov, P., Groefsema, H.: Requirements and tools for variability management. In: IEEE Workshop on Requirement Engineering for Services (REFS 2010). IEEE COMPSAC (2010)
10. Xu, L.: Monitoring Multi-Party Contracts for E-Business. PhD Thesis, Tilburg University (2004) ISBN 905668-128-1
11. Pesic, M.: Constraint-Based Workflow Management Systems: Shifting Control to Users. PhD thesis, Eindhoven University of Technology (2008)
12. Xu, L., de Vrieze, P.T., Phalp, K.T., Jeary, S., Liang, P.: Interoperable End User Process Modelling for Process Collaborative Manufacturing. International Journal of Computer Integrated Manufacturing (2012)
13. de Vrieze, P.T., Xu, L., Bouguettaya, A., Yang, J., Chen, J.: Building Enterprise Mashups. Future Generation Computer Systems 27(5), 637–642 (2010)