Formalizing Artifact-Centric Business Processes
Towards a Conformance Testing Approach

Hemza Merouani[1], Farid Mokhati[2] and Hassina Seridi-Bouchelaghem[3]
[1]*Departement of Mathematics and Computer Sciences, ReLa(CS)[2] Laboratory, University of Oum El-Bouaghi, Oum El-Bouaghi, Algeria*
[2]*Department of Mathematics and Computer Sciences, LAMIS Laboratory, University of Oum El-Bouaghi, Oum El-Bouaghi, Algeria*
[3]*Department of Computer Sciences, LabGed Laboratory, University of Annaba, Annaba, Algeria*

Keywords: Business Process Management, Data-Centric Business Processes, Business Artifact, Maude-Strategy Language, Formalization, Validation by Simulation, Conformance Testing.

Abstract: Recently, Artifact-Centric Business Processes have emerged as an approach in which processes are centred on data as a "first-class citizen". A key challenge faced by such processes is to develop effective mechanisms that support formal specification, validation and verification of their static and dynamic behaviours i.e., the data of interest and how they evolve. We present, in this paper, a novel approach that allows on one hand, formalizing Artifact-Centric Business Process Models described in UML as an executable formal specification in the Maude and its strategy language and, on the other hand, testing whether the implementation of such models is conformant to its specification using all possible scenarios that are described as Maude strategies. One of the main reasons for using Maude-Strategy language is due to its execution environment in which the use of a wide range of formal methods is facilitated.

1 INTRODUCTION

Business Process Management (BPM) is "the discipline that combines knowledge from information technology and knowledge from management sciences and applies this to operational business processes" (Weske, 2012). Business processes are the cornerstone of BPM; a business process (BP), by definition, is "a collection of activities that takes one or more kinds of input and creates an output that is of value for the customer" (Hammer, 1993). BPs occur in almost all organizations, such as schools, government agencies, business, hospitals, etc. and often in the form of routine tasks in the daily life such as shopping, banking, shipping and checking in/out books from libraries.

During the last two decades, BPM has received extensive attention due to its potential for significantly improving enterprise productivity and diminution costs. Being widely adopted by industry, both researchers and practitioners in the BPM community have focused only on studying control flow aspects that define how a BP is supposed to operate, but giving little importance (or none at all) to the information produced as a consequence of the process execution. A common drawback of such modelling notations (such as BPMN, EPCs, Petri Nets etc.) is being activity centric i.e., lacking the connection between the process and the data manipulated during its executions. This reflects also in the corresponding verification techniques, which often abstract away from the data component. This problem affects many contemporary process-aware information systems, incrementing the amount of redundancies and potential errors in the development phase (Solomakhin et al., 2013).

To tackle this problem, IBM introduced data-centric business process models that give data a foundational role in the context of business process design (Nigam and Caswell, 2003). In particular, such models are based around Business Artifacts (BA) and their lifecycles. BA, sometimes referred as a business record, "is a concrete, identifiable, self-describing piece of information through which business stakeholders add value to the business". They are key business-relevant objects that combine both data and behavioural properties that are used as

DOI: 10.5220/0004951803680374
In *Proceedings of the 16th International Conference on Enterprise Information Systems* (ICEIS-2014), pages 368-374
ISBN: 978-989-758-028-4

primitive driving the process modelling.

(Nigam and Caswell, 2003) defined operational specifications of a business artifact, and from there many research efforts, methodologies and meta-models have originated. Furthermore, the formal foundations of the artifact-centric paradigm are being investigated (Gerede *et al*., 2007) (Bhattacharya *et al*., 2007) in order to capture the relationship between processes and data and to support formal reasoning.

In fact, the lack of formalism and rigor in existing artifact-centric design models often leads to ambiguities and different interpretations. Those weaknesses combined with the inherent complexity of business processes management systems generate business processes without any rigorous conceptualization and many problems in their development process. Using formal notations to specify behaviour of artifact-centric business process models makes it possible to produce precise description. This also offers a better support to their verification and validation process.

In the setting of artifact-centric business process modelling, there are three levels in the specification of the solution for a given process:

- Definition of the data involved in the process,
- Identification of the basic actions that manipulate those data, and
- Specification of how those basic actions must be used by the process to reach the goal.

BALSA framework (Bhattacharya, 2009) is an artifact-centric design methodology which is based on the definition of a Business Operation Model (BOM). These latter are defined across four explicit inter-related but separable dimensions: (1) Business Artifacts, (2) Lifecycles, (3) Services, and (4) the Associations of services to artifacts. BOMs serve as basis for system implementation and they are used as input for the conceptual flow and workflow realization phases. More recently, (Estañol *et al*., 2013) identified the UML diagrams that can be used to represent a process from an artifact-centric perspective following the BALSA framework. The importance of their contribution lies in the fact that UML is a high-level, technology-independent standard in the world of conceptual modelling and, in our opinion, it can be automatically translated into Maude (Clavel *et al*., 2011) and the translation can be used for reasoning purposes.

Maude is based on a sound and complete logic called rewriting logic (Meseguer, 1992). The advantages of using Maude in this context are many. First of all, it supports concurrent object-oriented computation. These properties of rewriting logic make it an ideal framework to support business artifact formalization. Secondly, since Maude is based on (conditional) rewrite rules, it is very natural to express the evolution of an artifact from state to another. And, last but not least, Maude allows us the separation between the first two levels above, data and actions (i.e. business entities with lifecycles) by distinguishing at the logic level equations from rewriting rules. Furthermore, Maude strategy language (Martí-Oliet *et al*., 2009) completed Maude by a third level of strategies to control and determine the right sequencing of those actions.

In this paper, we advocate the feasibility and the interest of: (1) Formalizing both static and behavioural properties of BALSA framework described in UML as an executable formal specification with Maude and its strategy language (2) Testing whether the implementation of such models is conformant to its specification using all possible scenarios which are described as Maude strategies. We are concerned with conformance testing approach for its ability to identify problems in either finite or infinite state systems where the state space becomes too large for model-checking.

The remainder of this paper is organized as follows: In section 2, we give a general outline on the major related works. We briefly present, in section 3, the BALSA framework, Maude as well as the Maude Strategy language. Section 4 presents our approach. In section 5, we give some conclusions and future work directions.

2 RELATED WORKS

In recent years, modelling, specification and analysis of artifact-centric business processes have attracted a lot of attention from the research community.

From modelling and specification viewpoint, business process models are habitually the first interface between business managers and software engineers. Different formalisms and notations are used to represent the four elements in the BALSA framework, (i.e. business artifacts, lifecycles, services and associations). A first challenge in this perspective is to find: (1) a rich and flexible modelling notation that provides understanding and access to all facets of BALSA framework and, (2) an appropriate formal notations which allows producing rigorous and precise descriptions efficiently supporting verification and validation process. Table 1 gives a brief overview of some recent and important works that deal with data-centric business process modelling. In our work, we chose to combine the advantages of the graphical modelling notation UML defined in (Estañol *et al.*, 2013) and the formal specification language Maude-Strategy in a single technique.

From formal analysis perspective, little is understood about artifact-centric business process. In general, the verification problem is undecidable because model-checking technique in the presence of data as a "first class citizen" makes the set of possible states infinite (Hull, 2008) (Gerede et al., 2007) (Bhattacharya et al., 2007). In this paper, we are concerned with testing technique. Testing can be used for identifying errors in infinite state systems. After validating the generated formal specification written in the Maude-Strategy language, we can use it for testing the conformity of the implementation models to their specifications.

Table 1: Some formalism and notations used to represent the four elements of BALSA framework.

	Artifact	Lifecycle	Service	Association
Bhattacharya *et al.*, 2009	Entity-Relationship model (ER)	State Machine	Input, Output, Pre-condition & Effects	Event &Condition &Action (ECA) rule
Hariri *et al.*, 2011	Data Base schema	State Machine	Pre & Post-Condition	Condition &Action
Deutsch *et al.*, 2011	Attributes & variables	Variable	Pre & Post-Condition	Business Rule
Hull *et al.*, 2011	Guard-Stage-Milestone (GSM)	Guard-Stage-Milestone (GSM)	/	/
Lohmann & Wolf, 2014	Petri Nets	Petri Nets	/	Petri Nets
Estanol *et al.*, 2013	UML Class Diagram	UML State Machine	OCL Contract	UML Activity Diagram

3 BASIC CONCEPTS

Before presenting the technical details of our approach, we present in this section some fundamental notions used in this study.

3.1 BALSA Framework in UML

In this sub-section, we give a brief description of the four BALSA dimensions and theirs representation using UML diagrams. For more details see (Bhattacharya et al., 2009) (Hull et al., 2009) and (Estañol et al., 2013).

3.1.1 Business Artifacts

The information models of business artifacts are intended to hold all of the information needed in completing business process execution. A business artifact has an identity, a set of attributes and relationship with other artifacts. In UML, a class diagram is used to show the business entities and how they are related to each other, represented as classes and associations respectively. Each class has a series of attributes.

3.1.2 Business Artifact Lifecycle

Lifecycles represent key stages in the evolution of an artifact, from its creation to its final disposition and archiving. Macro-lifecycles is represented as UML state machine.

3.1.3 Service

Services are units of work that make changes to one or more business artifacts. Typically, several services are applied during each stage of the lifecycle of an artifact. Services are specified by means of an OCL operation contract which consists in a set of input parameters and output parameters, a pre-condition and a post-condition.

3.1.4 Association

Associations are used to relate services and artifacts from the micro-level lifecycle of artifact; the goal is to define the right sequencing of services execution. In UML, activity diagram is used for specifying each external event in state machine diagrams when each service is represented as an action, arrows and control nodes show the order in which actions have to be executed.

3.2 Maude and Its Strategy Language

Maude (Clavel et al., 2011) is a specification and programming language based on rewriting logic (Meseguer, 1992) which allows the description of concurrent systems, this type of logic unifies all formal models of concurrency. The rewriting rules are of the form RL: `[t] -> [t'] if C`, which indicates that, according to rule `RL`, term `t` becomes `t'` if a certain condition `C` is verified, the condition is also optional. Three types of modules are defined in Maude. Functional modules allow defining data types and their functions. System modules allow defining the dynamic behaviour of a system. This type of module augments the functional modules by introducing rewriting rules. Finally, object-oriented modules, offer a more appropriate syntax to describe the basic entities of the object paradigm.

Once the Maude specifications become executable, we must ensure that the rewriting process does not go in undesired directions and eventually terminates. Maude-Strategy language (Martí-Oliet et al., 2009) can be used to control how rules are applied to rewrite a term in an attempt to control the non-determinism in the execution process. Strategies are defined in a separate module and they defined how those basic rewriting rules must be used to reach the desired solution. Besides providing basic strategies through the use of rule labels, the strategy language permits combining these strategies into more complex ones using several combinators: concatenation (`;`), union (`|`), and iteration (`E*` for zero or more iterations and `E+` for one or more iterations). Additionally, there is the combinatory `orelse` is a typical if-then-else. Given a Maude system module `M`, the user can write one or more strategy modules to define strategies for `M`. Such strategy modules have the following form:

```
smod  STRAT  is
   protecting M .
   including STRATj .
   ...
   strat E1 : T11 ... T1m @ K1 .
   sd E1(P11,...,P1m)  := Exp1 .
   ...
   strat En : Tn1 ... Tnp @ Kn .
   csd En(Qn1,...,Qnp)  := Expn, if C
endsm
```

where `M` is the system module whose rewrites are being controlled, STRAT$_j$ are imported strategy sub-modules, E_1, ...,E_n are identifiers, and Exp$_1$,...,Exp$_n$ are strategy expressions. The strategy rewriting command is: `srew T using E`, which rewrites a term `T` using a strategy expression `E`.

4 OUR APPROACH

Significant research challenge for data-centric workflow is the integration of reasoning into various stages of the business processes lifecycle. Due to the lack of effective and efficient tools, business processes management systems models in practice are not designed using rigorous techniques nor analyzed with verifiers, this leads to many issues (Hull et al., 2009). Figure 1 illustrates the proposed approach. Our long-term goal in this work is to develop a novel methodology with sound tools to design, formalize, implement and testing the conformity of the implementation of such models to their formal specifications. The dashed red bold-line rectangle in the figure 1 correspond to the first steps for reaching this goal that we advocate in this section: (1) modelling, (2) formalizing the data-centric business processes models, (3) validating the generated formal specification and (4) generating the tests cases from the validated formal specification.

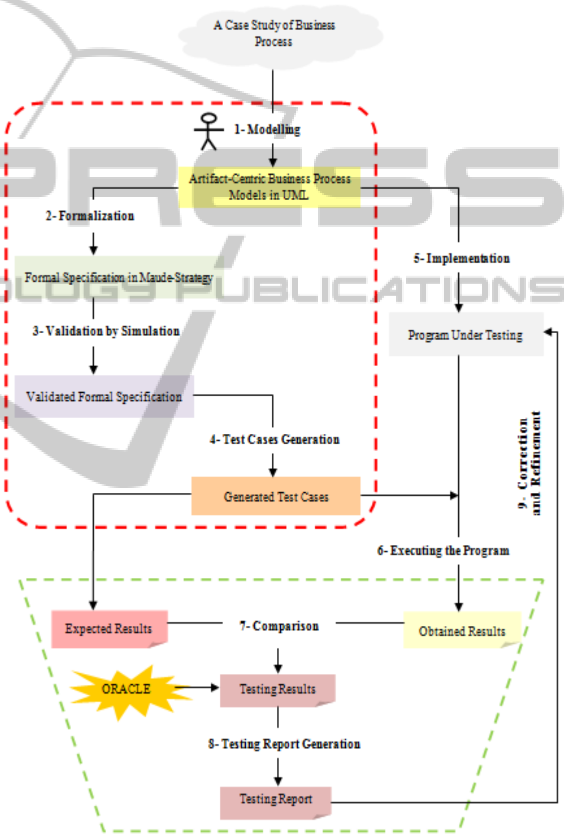
A Case Study of Business Process
1- Modelling
Artifact-Centric Business Process Models in UML
2- Formalization
Formal Specification in Maude-Strategy
3- Validation by Simulation
Validated Formal Specification
4- Test Cases Generation
Generated Test Cases
5- Implementation
Program Under Testing
9. Correction and Refinement
6- Executing the Program
Expected Results
7- Comparison
Obtained Results
ORACLE
Testing Results
8- Testing Report Generation
Testing Report

Figure 1: The methodologies of our approach.

4.1 Modelling

Business process models are habitually the first interface between business managers and software engineers. Partly due to the traditional division of academic disciplines it is often the case that in an application context these two groups of people have different technical backgrounds. As a result, business process models that are understandable and usable by one community are typically not understandable and usable by the other. This leads to significant communication problems, and a significant cost (Hull et al., 2009).

In our work, we chose UML to represent business process models. The graphical modelling notation UML is a high-level, technology-independent standard in the world of conceptual modelling. Outputs of this step are: (1) UML class diagram shows the business artifacts and how they are related to each other, (2) set of UML state machines represent the macro-level lifecycle of artifacts, (3) set of OCL contracts represent services and (4) a set of UML activity diagrams represent the micro-level lifecycle of artifacts.

4.2 Formalisation Process

In this section, we present the translation process in order to give a formal semantics of UML/OCL concepts generated in the first step using Maude and Maude strategy language. The table 2 summarize correspondences between the concepts abstracted from UML/OCL and Maude (strategy) language.

Thanks to the strong correspondence between UML class diagram concepts and the one in Maude language, the generation of Maude specification from UML class diagram is easily made. Every UML class and its attributes are formalized by a class with a set of attributes in object-oriented module of Maude. In addition, inheritance is directly supported by a subclass declaration.

To define UML state machine diagram, we propose the functional module `Machine-Diagram`. This module mentions all states and actions constituting the diagram that are defined in separated `STATE` and `ACTION` modules. Furthermore it includes the definition of two operations, `TargetState` that determines the state destination according to a state source and a condition, and the `AccomplishedAction` operation to determine the executed action according to a state and an event. The latter is formalized by a Maude operation `msg`.

Our way of representing OCL constraints is by means of (conditional or unconditional) rewrite rule `CRL` labelled with service name which express: input and output objects (i.e. instances of class), eventually a preconditions states the condition that must be true before executing the RL and the post-condition indicates which attributes change in certain objects after RL execution.

Since the goal of the activity diagram is to define the right sequencing of services execution i.e., control flows, we propose using several Maude strategy combinators for specifying them. Table 2 presents also the description of some elements of the UML activity diagram concepts and the corresponding formal semantics. Each task (service) is already represented as rewriting rule. In this way, sequence strategy shows the order in which `RLs` have to be executed. `OR-Fork` and `OR-Join` will be mapped using the conditional Maude strategy `orelse`. A specific condition (guard) of the conditional rewriting rules (`CRL`) determines which path (rewriting rule) will be taken (executed). `OR-Fork` (`ORJoin`) can be represented by a strategy which expresses the concatenation between the rewriting rule associated to the incoming segment (outgoing segment) and one of the conditional rewriting rules associated to the outgoing segments (incoming segments). We can translate `AND-Fork` and `AND-Join` in Maude strategy language using union combinator (`|`). `AND-Fork` is translated into a

Table 2: Mapping from UML/OCL concepts to Maude/Maude strategies specifications.

BALSA Concept	Class Diagram concepts		Maude Specifications
Business Artifact	Class		`class C`
	Attributes		`class C| a1 : S1, …,an : Sn`
	Inheritance Relation		`subclass C' < C`
BALSA Concept	**State Machine Diagram concepts**		**Maude Specifications**
Business Artifact Lifecycle	State		`(fmod STATE is ... endfm)`
	Transition	Event	`msg Evt :... -> Msg`
		Action	`(fmod ACTION is ... endfm)`
BALSA Concept	**OCL Concepts**		**Maude Specifications**
Service	Set of Input Parameters		crl [service-name] : < O_1 : C_1 / $ListeAt_1$ > ... < O_m : C_m / $ListeAt_m$ > => < O_1 : C_1 / $ListeAt'_1$ > ... < O_m : C_m / $ListeAt'_m$ > if <Condition> .
	Set of Output Parameters		
	Precondition		
	Post-Condition		
BALSA Concept	**Activity Diagram Concept**		**Maude Strategies Specifications**
Association	Task		`crl [service-name]`
	Control Nodes		sd sequence: = RL_1 ; RL_2 ; RL_3 sd ORFork:= RL_1;(CRL_2 orelse RL_N) sd ORJoin:=(CRL_1 orelse RL_N); RL_{N+1} sd ANDFork := RL_1 ; (RL_2|...|RL_N) sd ANDJoin :=(RL_1|RL_N); RL_{N+1} ....

strategy which expresses the execution of the rewriting rule associated to incoming segment followed by the concurrent execution (in Parallel) of rewriting rules associated to the outgoing segments.

4.3 Validation by Simulation

Since Maude and its Strategy language are supported by powerful rewrite engine (Clavel *et al*., 2011), the formal specification produced in the previous step benefits the access to the arsenal of generic tools for rewriting logic engine, such as simulation, LTL model checking, inductive theorem proving, etc.

The rewriting logic offers a great flexibility in terms of simulation of a specification, in particular, while choosing the initial configuration. Using all the system's description, we can validate a part of the system without involving the rest. Maude provides two commands for doing simulation: `rewrite` and `search`. The first command explores a possible execution path from an initial state to another one. However, the second one allows us to explore reachable state space in different ways.

4.4 Conformance Testing Process

Starting from a validated Maude-Strategy formal description of the Business process, the proposed method allows, in the first step, analyzing this description and extracting from it the possible testing sequences representing the different possible scenarios (i.e., the different strategies). These latter are analyzed in order to extract the possible test cases. Each test case contains input data and expected results. For testing the conformity between the implementation of the Business process and its formal specification, we proceed to: (1) execute the program under testing using the input data, (2) compare the obtained results to the expected ones using a test oracle. This latter represents a mechanism that is used during testing to determine whether software behaves correctly or not. (3) Finally, a testing report is generated for helping users to correct the potential errors (see figure 1).

5 CONCLUSIONS

In this paper, we have proposed a novel approach for formalizing, validating and testing the data-centric business process described in UML with the Maude-Strategy language. This work in progress represents, in fact, the first step towards developing an entire methodology supported by sound tools to various stages of the business processes lifecycle.

As future directions, we are working on the development of an environment supporting the proposed approach by using MDE (Model Driven Engineering) techniques for implementing the formalization process.

REFERENCES

Bhattacharya, K., Gerede, C., Hull, R., Liu, R., & Su, J. (2007). Towards formal analysis of artifact-centric business process models. In BPM (pp. 288-304). Springer Berlin Heidelberg.

Bhattacharya, K., Hull, R., & Su, J. (2009). A data-centric design methodology for business processes. Handbook of Research on Business Process Modeling, 503-531. IGI Global.

Clavel, M., Durán, F., Eker, S., Lincoln, P., Martı-Oliet, N., Meseguer, J., & Talcott, C. (2011). Maude manual (version 2.6). University of Illinois, 1(3), 4-6. USA.

Deutsch, A., Hull, R., Patrizi, F., & Vianu, V. (2009). Automatic verification of data-centric business processes. In Proceedings of the 12th ICDT (pp. 252-267). ACM.

Estañol, M., Queralt, A., Sancho, M. R., & Teniente, E. (2013). Artifact-Centric Business Process Models in UML. In BPM Workshops (pp. 292-303). Springer Berlin Heidelberg.

Gerede, C. E., Bhattacharya, K., & Su, J. (2007). Static analysis of business artifact-centric operational models. In SOCA (pp. 133-140). IEEE.

Hammer, M. (1993). Champy J. Re-engineering the corporation: a manifesto for business revolution. 1st edn. Harper Business, New York.

Hariri, B. B., Calvanese, D., De Giacomo, G., De Masellis, R., & Felli, P. (2011). Foundations of relational artifacts verification. In BPM (pp. 379-395). Springer Berlin Heidelberg.

Hull, R. (2008). Artifact-centric business process models: Brief survey of research results and challenges. In OTM (pp. 1152-1163). Springer Berlin Heidelberg.

Hull, R., Damaggio, E., Fournier, F., Gupta, M., Heath III, F. T., Hobson, S & Vaculin, R. (2011). Introducing the guard-stage-milestone approach for specifying business entity lifecycles. In WS-FM (pp. 1-24). Springer Berlin Heidelberg.

Hull, R., Su, J., co-chairs, (2009). Workshop on Data-Centric Workflows. Report of workshop Sponsored by NSF. Arlington Virginia, USA.

Lohmann, N., & Wolf, K. (2014). From Artifacts to Activities. In Web Services Foundations (pp. 109-135). Springer New York.

Martí-Oliet, N., Meseguer, J., & Verdejo, A. (2009). A rewriting semantics for Maude strategies. Electronic Notes in Theoretical Computer Science, 238(3), 227-247. Elsevier.

Meseguer, J. (1992). Conditional rewriting logic as a

unified model of concurrency. Theoretical computer science, 96(1), 73-155. Elsevier.
Nigam, A., & Caswell, N. S. (2003). Business artifacts: An approach to operational specification. IBM Systems Journal, 42(3), 428-445. NY, USA IBM.
Solomakhin, D., Montali, M., Tessaris, S., & De Masellis,R. (2013). Verification of Artifact-Centric Systems: Decidability and Modeling Issues.
Weske, M. (2012). Business process management: concepts, languages, architectures. Springer.