A Specific Language for Developing Business Process by Refinement based on BPMN 2.0

Salma Ayari[1], Yousra Bendali Hlaoui[1] and Leila Ben Ayed[2]
[1]*Faculty of Sciences of Tunis, University of Tunis El Manar, Tunisia*
[2]*National School of Computer Science, University of Mannouba, Tunisia*

Keywords: BPMN, Refinement, Context-free Grammar, Syntax-directed Translation, Formal Semantics, Formal Verification.

Abstract: This paper deals with the development by refinement of BPMN(Business Process Model and Notations) models. Indeed, Business Process (BP) development based on step-wise refinement **(i)** facilitates the understanding of complex BP **(ii)** specifies all the BP semantics that the BPMN model has to describe by focusing on the smallest detail of the BP since details are added gradually to the model under development.
Hence, we propose an approach assisting a business process developer to gradually build his/her BPMN model ensuring an automatic syntax and semantic property checking. The approach allows in one hand a BPMN syntax driven refinement based on a BPMN context-free grammar and in the other hand a formal verification of semantic properties. To be validated, the proposed approach is illustrated throughout the development of an on line flight booking BP.

1 INTRODUCTION

As BPMN (Specification, 2006) is a standard used for the modeling of business processes, we propose, in this paper, an approach for developing and verifying BPMN models based on automatic refinement process and formal verification techniques (Ayari et al., 2018). In this approach, business process development starts with an abstract BPMN sub-process satisfying certain semantic properties represented through its pre and post conditions. Hence, a BPMN sub-process $Sub-P$, is refined by a series of BPMN constructs R_{BP} which should **(i)** preserve the initial $Sub-P$ semantics by adding new semantics and **(ii)** be syntactically correct accordingly to BPMN syntax. Hence, at each level of the refinement, semantic properties and syntax rules have to be checked in order to provide reliable, valid and correct refined BPMN models. In fact, at each refinement level, developers add more relevant details with preserving semantic properties of the model they refine in order to ensure a correct by construction BPMN modeling. Therefore, developers focus more on the BP smallest semantic detail that the BPMN model should describe and represent. In addition, to maintain the BPMN syntactic correctness of refined models throughout the BP construction, we guide the developer in the modeling task by providing a set of BPMN refinement patterns. These patterns are defined via a formal framework to ensure the automation of the refinement process. Hence, we have developed a context-free grammar (Aho et al., 1986) which generates systematically refined models by choosing the kind of refinement pattern. The developer intervenes in the refinement process by choosing the right pattern to use, by defining the BP tasks and by specifying semantic requirements. Therefore, we ensure the development of syntactically correct BPMN models. Moreover, at each level of the refinement, we have to prove refinement properties which ensure semantic coherence between two linked refinement models. These properties define the semantic relationship that should exist between refined model and refinement model. To ensure the proof of the satisfiability of these properties, we use Event-B (Abrial, 2010) formal method as it is based on model refinement and allows the proof of its correctness. In addition, certain functional properties should be checked to ensure the reliability of developed BP models. In this aim, we use NuSMV model checker (Cimatti et al., 1999) which verify the satisfiability of these properties described by the CTL (Kumar et al., 2004) temporal logic. Thereby, formal BPMN model semantics are specified per a Kripke structure (Kripke, 2007) as we use an event based se-

Ayari, S., Hlaoui, Y. and Ben Ayed, L.
A Specific Language for Developing Business Process by Refinement based on BPMN 2.0.
DOI: 10.5220/0010561404890496
In *Proceedings of the 16th International Conference on Software Technologies (ICSOFT 2021)*, pages 489-496
ISBN: 978-989-758-523-4

mantics. This structure is ,systematically, provided by a Syntax-Driven Translation (SDT) (Aho et al., 1986) which is developed accordingly to the BPMN grammar. Over this structure, event-B verifies the preservation of semantic properties during the refinement process and NuSMV checks BP functional properties.

This paper is structured as follows: In the following section, we will explain the problem and propose our solution (Section related work), in the third section, we will define our approach of development by refinement and the formal verification of BPMN models. In the fourth section, we will introduce the notion of contracts based on the BPMN refinement. In the fifth section, we will define the formal semantic's of BPMN with different refinement patterns. In section 6, an example illustration will be explained. And finally, in section 7, a conclusion is completed.

2 RELATED WORK

Refinement based business process modeling is one of the key challenges that business process management systems must meet. However, most of the existing approach restrict business process refinement to the run-time. Furthermore, in our opinion, the refinement increases the business process quality not only during the run-time but also during its development. Nevertheless, we present and discuss the most important existing approaches in this framework. First, we discuss business process refinement approach second we discuss business process verification approaches. Then, we discuss an approach that combines business process modeling and verification. Kubovy et al. (Kubovy and Küng, 2014) provide a possible refinement of Business model and Notation (BPMN) Gateway activation concept for non-event based Gateway. A decomposition of business process model approach is proposed by Draheim in (Draheim, 2014). Wisniewski (Wiśniewski, 2017) carries out a decomposition of business process models into reusable sub-diagrams. Gol Mohammadi et al. (Mohammadi and Heisel, 2017) present a framework for systematic refinement of trustworthiness requirements to manage business processes. They discuss trust issues in the context of business process management using BPMN and i^*. Indeed, few work are regarding business process refinement during the development stage. Most of them use refinement to increase the business process reliability and efficiency during its running-time. However, the work (Wiśniewski, 2017) presents the refinement as a decomposition technique to provide reusable patterns to be used in further business process modeling tasks. The correctness of the integration of this patterns into business model is ensured manually and not systematically by the developer or the pattern user. Morales et al. (Mendoza Morales, 2014) propose a systematic vision of analysis, design and verification of business processes by incorporating the use of Timed Automata and model checking techniques. Brumbulli et al. (Bouchaala et al., 2014) provide a syntactic correctness to prevent the improper usage of the modeling elements.Dechsupa et al. (Dechsupa et al., 2018) propose a formal verification techniques for transformation of the BPMN model using a partitioning approach into Colored Petri Net (CPN) (Peterson et al., 1980). Bryans et al. (Bryans and Wei, 2010) describe a formal analysis of BPMN models using event-B. The described approaches can be considered as variations of either model checking (Nusmv,UPPAL,CPN,$\cdots$)Theorem proving event-B or simulations. In particular, model checking is the most often used to verify correctness problem expressed in LTL or CTL formula. Up to our best knowledge, no attempts has been made to formally define semantics of BPMN modeling language, perform verification for behavioral elements and prove the preservation of specified requirements during the transformation process from BPMN to formal or model checker languages. In this work we provide a BPMN refinement approach for modeling and verifying business processes. We use the refinement to increase business model quality as we focus the smallest detail in the modeling process. To perform correct refinement, we are brought about defining a formal framework in which we specify formally semantics for BPMN models. Furthermore, the semantic linkage between refinement levels is ensured through gluing properties which are checked automatically using event-B to maintain semantic model coherence. In addition, We consider a broad range of functional and behavioral properties to be checked on the provided model using NuSMV model checker. BPMN to NuSMV language transformation preserves semantic coherence by defining syntax directed translations (SDT) built on a context-free grammar that we have developed to ensure not only a syntactically correct BPMN refinement but also to provide refined model systematically.

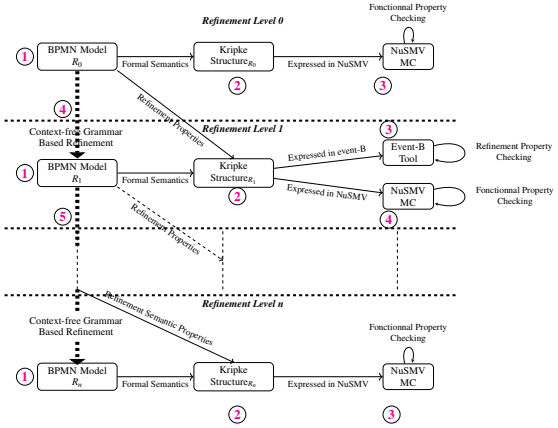


Figure 1: The proposed approach.

3 REFINEMENT BASED DEVELOPMENT AND FORMAL VERIFICATION OF BPMN MODEL APPROACH

We propose an approach for modeling BPs based on BPMN refinement and hierarchical formal verification. This approach guides developers to gradually build correct and reliable BPMN models. As Fig. 1 shows, a developer starts with specifying abstract BPMN model, $BPMN_{R0}$, which belongs to first level of refinement (*Level 0*). $BPMN_{R0}$ semantics is described via a set of properties in terms of *pre* and *post-conditions* of the BP. To formally check the reliability of $BPMN_{R0}$ model, we formalize its semantics over a Kripke structure (Kripke, 2007) model. Using CTL temporal Logic (Kumar et al., 2004) and based on semantic properties, we express functional properties which should be satisfied to verify $BPMN_{R0}$ reliability. Once specified, these properties are verified using NuSMV model checker (Cimatti et al., 1999). If the property is verified, the developer can refine the $BPMN_{R0}$ model otherwise, he/she has to correct $BPMN_{R0}$ model in order to satisfy the expected functional property. The refinement is decided by the developer and guided by the approach.

4 A BPMN REFINEMENT BASED ON CONTRACTS

In this paper, we use the notion of system refinement contracts(Le and Passerone, 2014) to ensure the correctness of BPMN refinement. Hence, we define (1)*business process refinement basic contract*, (2)*business process refinement behavioral contract* and (3)*business process refinement synchronization contract*. These contracts are used to relate and trace syntactic requirement mapping from the abstract business process to the refined one.

4.1 Basic Business Process Refinement Contract

Contract: *The refinement is based on the Business Process Modeling Notation (BPMN) language constructs.*

BPMN Specification. BPMN is a standard for modeling business processes flow proposed by the OMG (Object Management Group).

Definition 1. *(BPMN Specification) a business process BP is defined by: BP = (**Objects**, **Artifact**, **Swimlane**, **Connection**) is a business process with:*

- ***Objects*** $= O^{Event} \cup O^{Activity} \cup O^{Gateway}$ ***O*** *is a set of objects with:*
  1. $O^{Event} = \varepsilon^S \cup \varepsilon^I \cup \varepsilon^E$*, i.e.the set of events.*
  2. $O^{Activity} = O^{Task} \cup O^{sub-process}$*, i.e. the set of activities.*
  3. $O^{Gateway} = O^P \cup O^{Ex} \cup O^{In}$*, i.e. is the set of gateways.*
- ***Artifact*** *is the set of artifacts used to provide additional information. Also called notations, they serve as special labels for objects and arcs belonging to the BPMN process.*
- ***Swimlane*** *are rectangular boxes that represent the participants of a business process. A lane can contain flow objects that are executed by that path (participant).*
- ***Connection*** *is a set of objects defining the sequence flow, message flow or association defining the the sequencing order between different BPMN constructs defined above.*

A business process is initially grouped by a set of related BPMN constructs designed for the purpose of having an expected response from the process or producing a value. It is a process initiated by an initial event and terminated by an End event. Thus, the starting state of a business process is the initial event ε^S. Then a specified execution stream *Seq* is triggered from the start event to the *Sub-P* sub-process where it can be triggered. This sub-process must confirm certain conditions. These conditions are used for traceability purposes, so it is possible to check the different levels of refinement. When the requirements change, it is possible to know which sub-process might be affected. In this thesis, we study these conditions by

way of the pre- / post conditions which surround a sub-process (Pre_Sub-P and $Post\_Sub-P$), as as well as a transformation invariant, which defines a resulting refinement relation between the abstract and the refinement of a sub-process.

4.2 Behavioral Refinement Contract

contract: This contract defines the behavior of refinement and not the behavior of the business process. Therefore, in a contract, we define a precondition which, when satisfied, the refinement could be performed. Further, in this contract, we determine how to perform the refinement via a refinement invariant and specify the effect of the refinement on the business process through a post-condition that must be satisfied after the refinement. ***contract:*** The refinement is carried out accordingly for:

1. **pre-condition:** The refinement is applied only on the construction of BPMN sub-processes.
2. **Invariant:** The refinement is performed according to the BPMN syntax. We distinguish between different types of refinement, namely, sequence refinement, parallel refinement, exclusive refinement and iterative refinement. For each type of refinement, we determine an invariant like a BPMN pattern. Formally, we define a refinement invariant as a rule for producing a context-free grammar specifying the BPMN syntax. Therefore, each refinement invariant is syntactically correct.
3. **Post-condition:** is the effect of the refinement after having satisfied the invariant. Indeed, the satisfaction of the invariant is the introduction of a relative BPMN refinement model in the initial business process model. Post-conditions are specified after analyzing the impact of the change on the BPMN model after refinement. In the next section, we establish this analysis.

A *Sub-P* sub-process is an abstract activity that refines a collection of other tasks or / and sub-processes. It specifies different scenarios, depending on the prerequisites and afterwards. Since each sub-process is linked to some kind of pattern namely sequence pattern *SeqRef*, parallel pattern*ParRef*, exclusive pattern *ExcRef* or loop pattern *LoopRef* depending on a Pre_SeqP and a $Post\_SeqP$, Pre_ParP and a $Post\_ParP$, Pre_ExcP and $Post\_ExcP$ or Pre_LoopP and $Post\_LoopP$ respectively for each pattern. Each refinement model can initially be triggered by the start event and end its execution with the end event. A sub-process can also be refined into a simple task which is an atomic task in which it looks like it could be the last level of refinement. A simple*S-Task* depends on its Pre_S-Task and $Post\_S-Task$ start with the start event and end with the end event.

4.3 Synchronization Contract

contract: Which is carried out to analyze the effect of the impact of change and ensured by the property link. This includes the synchronization of the refinement according to the overall model. The dependence between the activity under the BPMN process and the different refinement models is represented by each refinement pattern (*SeqRef*, *ParRef*, *ExcRef* or *LoopRef*) can be presented by a sequence of sub-processes, sub-processes which behave in a parallel manner, an alternate execution flow of the sub-processes according to system state conditions or via an iterative task called a Task *L-Task* loop respectively.

These contracts ensure that the link between the abstract model and the refinement levels containing the contracts correctly assembles and satisfies the requirements. According to these requirements, we are about to build BPMN processes using refinement by formalization requirements adopted at each refinement level in the form of contracts.

The grammar can be used as contracts that must ensure consistency in terms of modeling BPMN processes using refinement patterns. By a contract, the developer, according to the result expected by a contract, he can choose the patterns to use. A contract consists of an input, verifying a certain syntax, and transformation rules that guarantee a specific result. The contracts we introduce are presented in (Ayari et al., 2019) in the form of non-contextual grammar.

5 SEMANTIC DESCRIPTIONS OF BPMN WITH DIFFERENT REFINEMENT PATTERNS

The semantic description is the behavior of the business process described by the instantiation function of the flow Object (O) and more precisely of the activity (task or sub-process) which shows the steps performed in a process. The behavior is presented by a sequence of allowed states $\{start-subP, finishsub-P\} \in S$. The formal semantics of an activity belonging to a BPMN process is defined as follows:
The Sub-P activity is triggered by the transition (**idle-subP**), which allows to go to the *begin event* event to order the sub-P activity to execute and goes into trigger state. After the activity goes through the transition (**start-subP**), its *terminate event* occurs and the tran-

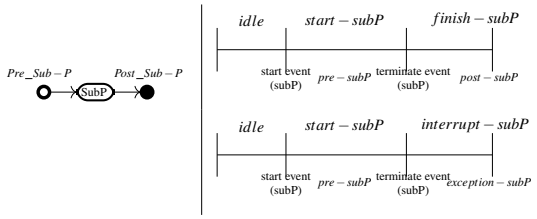


Figure 2: BPMN semantic activity.

sition (**finish-subP)**) is executed. The state *start-subP* is labeled with the precondition *pre-subP* and the state *finish-subP* is labeled with the post-condition *post-subP*(see figure 2). Another execution scenario is possible when executing a BPMN sub-process. When the activity is in active state, it may be waiting for the termination event but the system cannot respond to this emitted event which may be caused by timed out or other interrupt. Thus, the *interrupt-subP* event is activated just after the start event *start-subP*. The order of execution of a BPMN process is shown in figure 2.

A BPMN sub-process is decomposed into a set of activities following one of the refinement patterns presented in the previous chapter. It is refined, following syntactic rules introduced through non-contextual grammars, into a set of activities linked together by a pattern. A BPMN sub-process invokes an activity whose behavior is described in another BPMN process (which refers to a business sub-process).
A BPMN process can contain a sequence pattern, a parallel pattern, an exclusive pattern or an iterative pattern. In our approach a BPMN sub-process belonging to an abstraction level *i* can be refined (decomposed) by other activities belonging to the level *i + 1*.

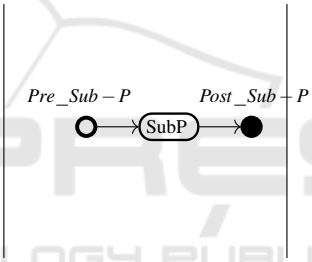
Summary of the Properties to Verify. This section summarizes the systematic properties defining exactly what needs to be proven for this BPMN sub-process to perform properly (see figure 3).

$$idle \vDash \neg start - event(subP)$$
$$start - subP \vDash start - event(subP) \wedge pre - subP$$
$$finish - subP \vDash end - event(subP) \wedge post - subP \wedge (pre - subP \Rightarrow post - subP)$$
$$interrupt - subP \vDash exception - subP$$

Figure 3: Properties to check for a BPMN sub-process.

Although the formal semantics of BPMN processes follows that of Petri nets, we will adopt the semantics of Kripke structures proposed for development of BPMN processes. This is explained by the fact that our business process system is a state / event driven system whose behavior is seen as a set of states and state transitions due to the occurrence of events. Indeed, the semantics of this behavior is equivalent to that of Kripke structures. From a BPMN model we generate a model of the Kripke structure representing the denotation semantics of the starting model. In order to provide generic semantics, relative to the refinement process, we generated for each BPMN refinement pattern, a model of the Kripke structure. We illustrate this generation through the ***sequence*** and ***exclusive pattern***. The semantics of BPMN processes with specified by refinement patterns is defined in terms of Kripke structure. We will also focus on the refinement properties of the BPMN model. We will give their definitions, then explain how the required properties are described from the system behavior constructs of BPMN models with refinement and how they can be formally stated. The table 1 gives a semantic description of a BPMN sub-process named A refined using Kripke structure.

Table 1: Semantics adopted for a refined sub-process A.

BPMN Object	Kripke structure
	
$P_{i-1} \hat{=} (Pre(A) \rightarrow Post(A)) \Rightarrow$ *Functional property*	
$P_{Ref_i} \hat{=} \emptyset \Rightarrow$ *Refinement property*	

5.1 Semantic Description of a BPMN Sequence Refinement Pattern

The concept of refinement offered by BPMN is defined in our approach following a specific pattern applied to BPMN process refinement. Figure 4 describes the semantics of the sequential refinement pattern for BPMN processes. The pattern is composed of two activities starting from the sequential behavior.
Each activity is labeled by the pre- and post-condition assertions. When the process enters a relative state of an invocation of the process with (**idle**), which allows to go to the *start event* to order the activity *sub-P01* to run and go into the trigger state. Once the activity goes through the transition **start-SeqP** and if nothing unexpected happens, **intermediate-SeqP** is triggered. The *termination event* occurs and the transition (**finish-SeqP**)) is executed. The state *start-SeqP* is labeled with the precondition *pre-SeqP*, The state

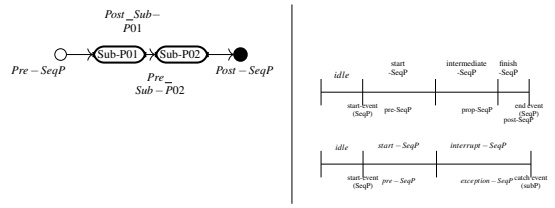


Figure 4: Semantics of a sequential BPMN refinement pattern.

$idle \vDash \neg start - event(SeqP)$

$start - SeqP \vDash start - event(SeqP) \wedge pre - subP01 \wedge pre - subP \Rightarrow pre - SeqP$

$intermediate - SeqP \vDash post - subP01 \wedge pre - subP02 \wedge pre - subP02 \Rightarrow post - subP01$

$finish - SeqP \vDash end - event(SeqP) \wedge post - SeqP \wedge (post - SeqP \Rightarrow post - subP)$

$interrupt - SeqP \vDash exception - SeqP$

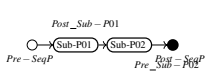
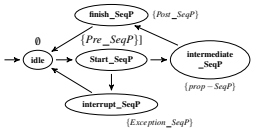
Figure 5: Properties to check for a BPMN sequence refinement pattern.

intermediate-SeqP is labeled with the assertion *prop-SeqP* and the state *finish-subP* is labeled with the post-condition *post-SeqP*. Figure 4 shows the formal semantics of a BPMN sequence refinement pattern.

Summary on the Properties to Check. The sequence refinement pattern must ensure that the sequence of activities $O_i^{Activity}(i \in \{1 \cdots n\})$. Here we have the activities *Sub-P01* and *Sub-P02* which refines the BPMN sub-process *SubP*. A sequence refinement requires proving the following properties (see figure 5).

Table 2 gives a semantic description of a BPMN refinement with the sequence refinement pattern $(>, \{B, C\})$ using the Kripke structure.

Table 2: Semantics adopted for the sequence refinement pattern.

BPMN objects	Kripke structure
	

$$P_{Ref_i} \hat{=} (Pre - subP \Rightarrow Pre\_SeqP) \wedge (Post\_SeqP \Rightarrow Post - subP)$$

$$P_{Fonct_i} \hat{=} (Pre\_SeqP \Rightarrow Post\_SeqP) \wedge Pos\_SubP01 \Rightarrow Pre\_SubP02$$

For lake of space, we have cited only the sequence refinement pattern in this paper.

6 EXAMPLE ILLUSTRATION

For the application of the refinement modeling process to BPMN processes, we introduce the example of online flight booking (Lam, 2010a), modeled in BPMN. We will focus on the *Client(Customer)* swimlane. The customer first enters the trip details, scrolls through the flight information and decides if they want to view more information about the flight.

6.1 Formal Verification with NuSMV

Figure 6 presents the development of *Travel Agency* business process (Lam, 2010b) via the tool that we have developed to support our BPMN refinement based modeling approach. As figure 6 shows, the developer starts the modeling activity by specifying *Travel Agency* `Sub_Process` which belongs to the first level of refinement (level 0). The developer interacts with the tool to specify the type of the BPMN refinement pattern to use for refining `Sub_Process` and indicates the number of activities composing the pattern. In this case, the developer use a refinement sequence pattern composed of ***Check-flight*** and ***Continue reservation***. At each level of refinement, the developer describes ***pre*** and ***post conditions*** specifying the semantic properties of the model. Accordingly to figure 3, the tool displays the formal semantics of *Travel Agency* `Sub_Process` and verifies automatically the specified functional property using NuSMV Model Checker:

$$\texttt{CTLSPECAG}(\texttt{TA1.reservationTerminate} \Rightarrow \texttt{TA.reservationdone})$$

which specify the safety functional condition meaning that the reservation is terminate if the process continue the reservation.

6.2 Formal Verification using the Event-B

At the second level of refinement (Level 1), in addition to the functional property, the tool checks refinement properties. These properties are defined following a study of BPMN refinement change impact that we have detailed in (Hlaoui et al., 2018). To prove refinement properties relating check flight and Continue Reservation sequence pattern to Travel agency sub-process, the tool checks the CTL formula

$$\texttt{AG}((\texttt{TA1.requestTravelRef} \Rightarrow \texttt{TA.requestTravel}) \Rightarrow (\texttt{AF}(\texttt{TA.reservationDone}) \Rightarrow (\texttt{TA1.reservationTerminate})))$$

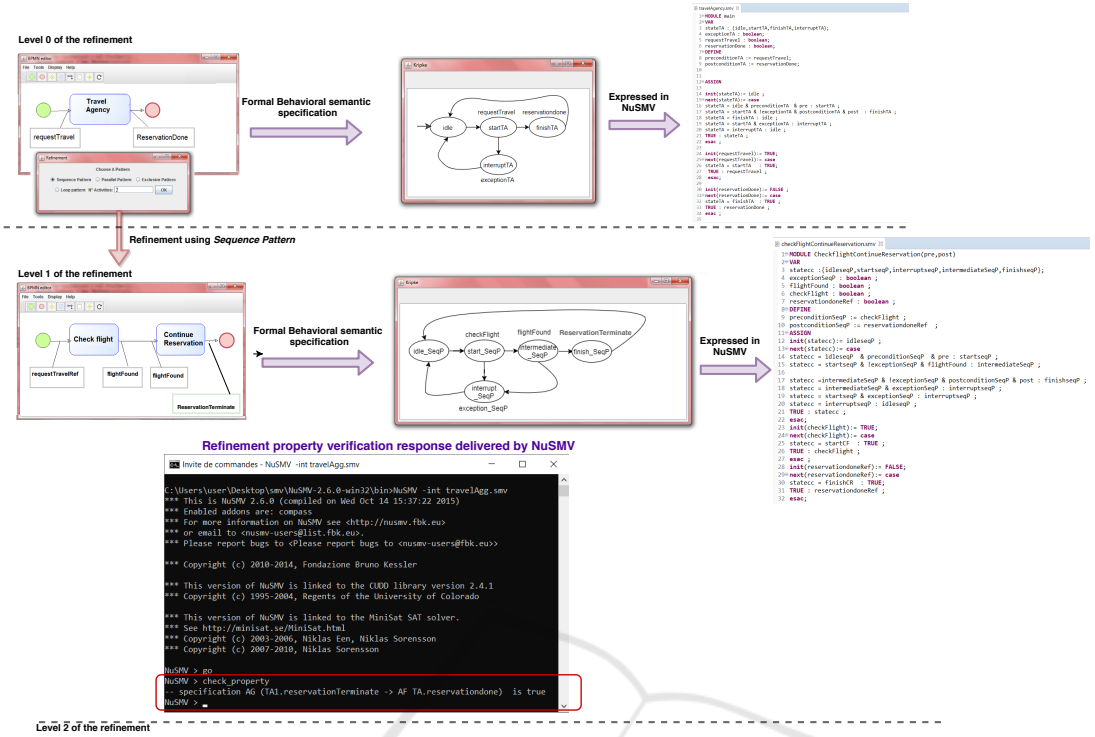


Figure 6: Travel Agency Business Process Development Based on BPMN Refinement verification using NuSMV.

stating that always if the pre-condition of ***check flight***, *requestTravelRef*, is true, the pre-condition of ***Travel Agency*** has to be true too and in the future the post-condition of ***Continue Reservation***,*reservationTerminate* has to be satisfied to verify positively the post condition of ***Travel Agency***, *reservationDone*. This formula is checked **True** within 2.68 sec. Therefore, the developer could continue the refinement task of the BPMN model.

7 CONCLUSION

To get a manageable complexity of workflow modeling, we have introduced a BPMN modeling refinement approach. This approach allows developers to be more detailed in the BPMN models at each level of refinement. In fact, we have proposed an automatic refinement based on a BPMN refinement pattern. We have introduced a set of rewriting rules called production rules to define the refinement at each level of abstraction. We defined a formal syntactic model for BPMN processes using the context-free grammar formalism and a formal semantic model through the Kripke Structure. Based on this structure, we check functional and refinement properties of gradually developed BPMN models using NuSMV model based on a hierarchical verification. Our future work is finalizing the tool with the introduction of a quantitative basis for design, development, validation and analysis of business process models.

REFERENCES

Abrial, J.-R. (2010). *Modeling in Event-B: system and software engineering*. Cambridge University Press.

Aho, A. V., Sethi, R., and Ullman, J. D. (1986). Compilers, principles, techniques. *Addison wesley*, 7(8):9.

Ayari, S., Hlaoui, Y. B., and Ayed, L. B. (2019). A grammar based approach to bpmn model semantic preservation using refinement. In *2019 IEEE 43rd Annual Computer Software and Applications Conference (COMPSAC)*, volume 2, pages 549–554. IEEE.

Ayari, S., Hlaoui, Y. B., and Ayed, L. J. B. (2018). A refinement based verification approach of bpmn models using nusmv. In *ICSOFT*, pages 563–574.

Bouchaala, O., Yangui, M., Tata, S., and Jmaiel, M. (2014). Dat: Dependency analysis tool for service based business processes. In *2014 IEEE 28th International Conference on Advanced Information Networking and Applications*, pages 621–628. IEEE.

Bryans, J. W. and Wei, W. (2010). Formal analysis of bpmn

models using event-b. In *International Workshop on Formal Methods for Industrial Critical Systems*, pages 33–49. Springer.

Cimatti, A., Clarke, E., Giunchiglia, F., and Roveri, M. (1999). Nusmv: A new symbolic model verifier. In *International conference on computer aided verification*, pages 495–499. Springer.

Dechsupa, C., Vatanawood, W., and Thongtak, A. (2018). Transformation of the bpmn design model into a colored petri net using the partitioning approach. *IEEE Access*, 6:38421–38436.

Draheim, D. (2014). On the trade-off between flexibility and extensionality in the decomposition of business process models. In *Novel Methods and Technologies for Enterprise Information Systems*, pages 63–77. Springer.

Hlaoui, Y. B., Ayari, S., and Ayed, L. J. B. (2018). Towards an automatic verification of bpmn model semantic preservation during a refinement process. In *International Conference on Software Technologies*, pages 397–420. Springer.

Kripke, S. (2007). Semantical considerations of the modal logic.

Kubovy, J. and Küng, J. (2014). Refinement of bpmn 2.0 inclusive and complex gateway activation concept towards process engine. In *Novel Methods and Technologies for Enterprise Information Systems*, pages 55–62. Springer.

Kumar, M., Schwiebert, L., and Brockmeyer, M. (2004). Efficient data aggregation middleware for wireless sensor networks. In *2004 IEEE international conference on mobile ad-hoc and sensor systems (IEEE cat. no. 04EX975)*, pages 579–581. IEEE.

Lam, V. S. (2010a). Formal analysis of bpmn models: a nusmv-based approach. *International Journal of Software Engineering and Knowledge Engineering*, 20(07):987–1023.

Lam, V. S. (2010b). Formal analysis of bpmn models: a nusmv-based approach. *International Journal of Software Engineering and Knowledge Engineering*, 20(07):987–1023.

Le, T. T. H. and Passerone, R. (2014). Refinement-based synthesis of correct contract model decompositions. In *2014 Twelfth ACM/IEEE Conference on Formal Methods and Models for Codesign (MEMOCODE)*, pages 134–143. IEEE.

Mendoza Morales, L. E. (2014). Business process veri cation: The application of model checking and timed automata. *CLEI Electronic Journal*, 17(2):3–3.

Mohammadi, N. G. and Heisel, M. (2017). A framework for systematic refinement of trustworthiness requirements. *Information*, 8(2):46.

Peterson, J. L. et al. (1980). A note on colored petri nets. *Inf. Process. Lett.*, 11(1):40–43.

Specification, O. F. A. (2006). Business process modeling notation specification.

Wiśniewski, P. (2017). Decomposition of business process models into reusable sub-diagrams. In *ITM Web of Conferences*, volume 15, page 01002. EDP Sciences.