A TOKEN BASED TRANSLATOR TO CONVERT BPMN TO SOLIDITY SMART CONTRACTS

Govinnage Rasika Perera

(168250J)

Degree of Master of Science

Department of Computer Science and Engineering

University of Moratuwa
Sri Lanka

April 2020

A TOKEN BASED TRANSLATOR TO CONVERT BPMN TO SOLIDITY SMART CONTRACTS

Govinnage Rasika Perera

(168250J)

Thesis submitted in partial fulfillment of the requirements for the degree
Master of Science in Computer Science

Department of Computer Science and Engineering

University of Moratuwa
Sri Lanka

April 2020

Declaration

I declare that this is my own work and this dissertation does not incorporate without acknowledgement any material previously submitted for degree or Diploma in any other University or institute of higher learning and to the best of my knowledge and belief it does not contain any material previously published or written by another per-son except where the acknowledgement is made in the text.

Also, I hereby grant to University of Moratuwa the non-exclusive right to reproduce and distribute my dissertation, in whole or in part in print, electronic or other medium. I retain the right to use this content in whole or part in future works (such as articles or books).

Signature: ……………… Date: ……………….
Name: G R Perera

The supervisor/s should certify the thesis/dissertation with the following declaration.
I certify that the declaration above by the candidate is true to the best of my knowledge and that this report is acceptable for evaluation for the MSc Research Thesis.

Signature of the supervisor: ……………………….. Date: ………………..
Name: Dr. Indika Perera

Acknowledgement

I would like to place record my deepest gratitude to Dr. Indika Perera in Department of Computer Science and Engineering, University of Moratuwa for the generous and invaluable guidance, suggestions and help provided to fulfil this research. Allowing me to experiment and explore my own and steps are altered whenever needed is a great fortunate I received. Thus, I would like to appreciate the guidance and allocating his precious time for my research throughout the whole time.

Also, I would like to thank Dr. Charith Chitraranjan, Department of Computer Science and Engineering, University of Moratuwa for co-ordinating the research and allocating his time for these works.

Last, but definitely not the least I wish to extend my sincere appreciation to Mrs. D. N. Perera (My Wife) for her inspiration and inputs for this research and all the hardships she went through to make this research a success. I have no valuable words to express my thanks for the all those people who helped me in this research in various aspects to make this dissertation possible.

Table of Contents

List of Figures

List of Tables

List of Abbreviations

BPM	:	Business Process Model
BPMN	:	Business Process Modelling Notation
XML	:	Extensible Markup Language
XSD	:	XML Schema Definition
AST	:	Abstract Syntax Tree
PoC	:	Proof Of Concept
SLOC	:	Source Lines of Code
LLOC	:	Logic Lines of Code

List of Appendices

1. INTRODUCTION

Contracts have been existed from the day humankind began selling goods and services. A common definition for a **contract** is "*a promise or set of promises for the breach of which the law gives a remedy or the performance of which law in some way recognizes as a duty*" [1]. It is a legal bond amidst two or more parties. Even in a single contract, there might be several agreements and transactions associated with it.

Due to the industrial revolution and globalization, contracts are getting smart. Conventional methods of dealing with business contracts have been changed enormously. One such technological enhancement is known as **blockchain** based **smart contracts**. Blockchain is a disrupting technology that enforces a record-keeping convention. Further, it can be perceived as a huge ledger for the financial transactions which majority agrees. Blockchains can be utilized as a ledger to record anything in the form of digital data which considered to be valuable and significant to the parties involved.

On the other hand, smart contracts are structured and developed to be executed on top of the blockchains. Thus, smart contracts inherit most properties and behaviours from the blockchain. In particular, properties such as immutability and distributed storage signifiers the uniqueness compared to traditional business contracts [2]. These features permit smart contracts to be comparatively significant in credible means when performing business contracts and business transactions out of it.

Smart contracts are perfectly matched for the enterprise domains such as banking, insurance, logistics, properties and fleet management…etc. in which the contracts can be automated using a sequence of specific set of rules and quantifiable terms [3] with the self-executable and self-enforceable features.

Further, smart-contracts have become a trending technology and most organizations are evaluating the use of smart-contracts to automate their business transactions which also provides openness to its stakeholders. These virtual agreements can be used in terms of exchanging valuable items such as money, content, shares, properties or any other significant material. The provided algorithm for the smart contracts works in a logical way ensuring that all the terms and conditions are satisfied. When considering the implementation, most common way of writing smart contracts is using the **Solidity** programming language. Solidity is a contract-oriented programming language that allows writing smart contracts for the blockchains [4].

There are noticeable inefficiencies when comparing traditional business transactions with the smart contracts. One such major issue is the intervention of an external third-party entity in the business operations. A trusted third party is required to impose the settlements and a cost is associated with such an involvement. Other limitations are such as delays in payments from other parties, difficulty in monitoring and managing each and every transaction in reality. Due to these inefficiencies, organizations are trying to move away from the traditional transactions.

However, it is worth to mention that even though there's a significant motivation of using smart contracts instead of traditional contracts there's limiting factors of this movement due to the nature of the contracts. In reality, there can be practical issues of applying such smart contracts and there can be situations that still requires a third-party to satisfy legal requirements [5].

Business Process Modelling Notation (BPMN) is a modelling technique to model a sequence of steps of a well-defined process. BPMN allows illustrating a set of business transactions and its associated flows of information in a graphical way. Moreover, this provides ability to perceive all business processes and represent their business procedures in a graphical and standard manner. Further, BPMN has a standardized specification and can be utilized as a bridge which connects the process motive and process development providing supplementary information and explicitness for the use of business process engineering [6]. Thus, it is evident that the BPMN is a useful notation in planning the business contracts and standardizing the business processes associated with it.

On the other hand, smart contracts implementation using Solidity requires a significant knowledge in Solidity programming language. If there's a possibility or a tool for converting outcome of the BPMN diagrams into Solidity smart contracts it will be truly beneficial for the business and community as well. Introduction of a such a translator will inherit the translation complexity as much as possible in order to provide a smooth experience in generating Solidity based smart contracts. Thus, a user with average knowledge in programming should be able to implement smart contracts with the help of a such BPMN process models to Solidity translator.

Therefore, major significance of this study is to evaluating the possibility of generating Solidity smart contracts from the outcome of a given BPMN diagram with the least expertise in Solidity programming language.

1.1. Problem Statement

Most of the available translators (or equivalent systems) addresses merely a subsection of BPMN elements or attempts to provide extensions for the already existing BPMN standard specifications [7]. Therefore, the users are not being able to convert process models in BPMN into Solidity smart contracts in a reliable manner.

Therefore, the main problem addressed in this research is defined as *"lack of an efficient translator to reliably convert business process models defined in BPMN into Solidity smart contracts"*.

Translator should be comparatively efficient when converting the BPMN diagrams into Solidity smart contracts. In other terms, a minimum usage of Ethereum gas and time units are being considered and it requires a simplified yet minimum number of lines of code as well.

An accurate translator must produce an executable Solidity code for a provided BPMN illustration of a valid business logic. Generated Solidity should be able to compile and deploy into the blockchain network with no compilation errors.

However, most of the existing solutions are considering a simplicity and ease of the process modelling task (design phase of the contracts) and very few has paid attention on reducing the cost of the generated contracts (deployment phase of the contracts). The deployment cost of the transaction is measured in Ethereum gas. For instance, the standard cost a transaction is 10gwei which is equivalent to 0.031USD in mainnet[1] network of Ethereum as of 08-12-2019.

Moreover, the readability of the Solidity code generated by already existing systems is quite law since it is difficult to associate and traceback the code blocks generated for particular business process elements.

In order to solve this problem, research question of this research is formulated as "*what would be the efficiency of a translation of business process models defined in BPMN in to Solidity smart contract code, produced by a drag-and-drop icon-based translator?*"

1.2. Goals and Objectives

The main goal of the research is *"to develop an efficient drag-and-drop icon-based translator to convert business process models defined in BPMN into Solidity smart contracts".*

In order to achieve this goal following objectives are defined;

1. Select a standard set of BPMN process elements that covers majority of business transactions
2. Map outcome of the (1) the selected BPMN process elements into Solidity language constructs
3. Develop the BPMN-to-Solidity translator from the outcome of (2)
4. Evaluate the efficiency and correctness of translation of the BPMN to Solidity smart contract code
5. Compare the translator with the existing systems

1.3. Scope and Limitations

Developing a translator to convert business process models defined in BPMN into Solidity smart contracts is the main focus of this research. Due to the time limitations, scope has been narrowed down and few assumptions are made.

The business process modelling technique being considered here, Business Process Modelling Notation (BPMN) has several versions and for the

[1] The cost of a standard transaction in Etherium mainnet is 10 gwei costing around 0.031USD https://ethgasstation.info/ online, accessed 08.12.2019.

simplicity BPMN version 2.0 which is managed by OMG group is considered.

A particular business process model defined in BPMN is assumed to be convertible if and only if the diagram is prepared using a valid set of BPMN elements defined in BPMN 2.0 specification with a valid business logic.

Another such assumption is that the user of this translator has an average knowledge in programming and BPMN. All the business process elements defined in BPMN 2.0 specification is not handled and it assumed that it covers basic control flow patterns which is sufficiently covers majority of the business use cases.

A translator will only cover the generation of the Solidity code and deployment of such Solidity smart contracts requires manual processes. Primary goal of the study is to construct a translator to assist creating smart contracts and does not substitute the manual process of generating smart contracts and is not targeted to be used as a non-technical material for creating such contracts. Performance and security requirements of the generated code still requires manual inspection.

1.4. Research Contributions

Following are the major contributions of this research.

i. A comprehensive study on related work and state-of-the-art techniques
ii. A collection of mapping principles and concepts that guides the translation of a business process models defined in BPMN to Solidity smart contracts
iii. A BPMN XML to Solidity smart contracts translator

2. LITERATURE REVIEW

In the introduction chapter, the research problem and significance of the research discussed. In this chapter, a comprehensive study focuses on identifying the key business process modelling techniques and tools which facilitates them are evaluated. It also provides an extensive review on already existing translators (or similar systems) for the BPMN to Solidity smart contracts translation. In addition, brief introduction to the smart contracts and BPMN 2.0 specification and Solidity language constructs discussed (Figure 2.1).

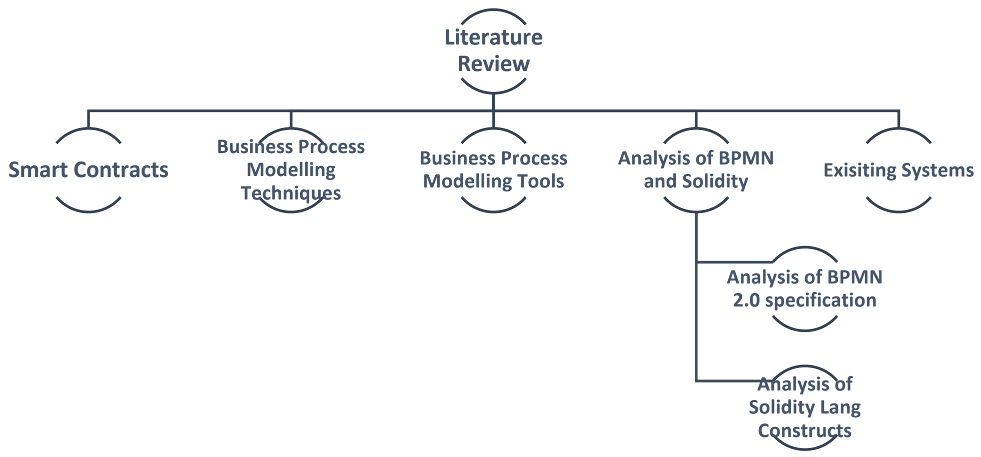

Literature Review
Smart Contracts
Business Process Modelling Techniques
Business Process Modelling Tools
Analysis of BPMN and Solidity
Exisiting Systems
Analysis of BPMN 2.0 specification
Analysis of Solidity Lang Constructs

Figure 2.1: Structure of the literature review

2.1. A brief introduction to the Smart Contracts

Smart contracts are considered as virtual contracts with self-executing and self-enforcing features managed by a specific set of terms and conditions defined. These contracts can be used in terms of exchanging valuable items such as currency, content & media, shares, properties or any other significant material. An example of an application of a smart contract is crowd sale. A person who is willing to start a new business can request capital in small amounts through a smart contract. Another example is trade receivable to trade payable contract. A particular transaction will check the balance of the trade receivable, if there is sufficient balance deal happens, or else the transaction block is invalidated. In this case, agreement of the contract is whether the trade receivable has sufficient money.

The obligations of the smart contracts are enforced automatically when the defined agreements are fulfilled. Once deployed, smart contracts could be performed absence of any person's intervention by gaining time and any extra effort [8]. When considering the applicability of smart-contracts, any field driven through the data such as insurance, logistics, banking, real-estate and fleet management…etc can be benefitted from self-executable nature of these smart contracts [3].

Business logic or the rules set which governs the business contract is encoded into smart contract with a programming language such as Solidity programming language. Upon deployment of the contract, these business rules are embedded into blockchain as machine instructions and are executed whenever a transaction occurs for the contract [9].

2.2. Business Process Modelling (BPM) Techniques Analysis

In this analysis, seven most common business process modelling techniques have been compared and analysed for the purpose of this research. Namely, (i) Business Use Cases (ii) Role Activity Diagrams (RAD) (iii) Data-Flow Diagrams (DFD) (iv) Flow Charts (v) Business Process Modelling Notation (BPMN) (vi) Petri Nets and (vii) Business Object Interaction (BOID) diagrams are evaluated.

The criterion used for the analysis is depicted in Table 2.1. These criterions were selected from a research by L. Aldin and S. de Cesare [10].

Table 2.1: Criterion for the Business Process Modelling Techniques Comparison [10]

Criteria	Description
Flexibility	The extent to which changes can be applied to a business process, only to those parts that need to be changed without replacing it completely.
Ease of Use	The extent to which how this technique can be applied without a specialized knowledge.
Understandability	The extent to which the technique can be understood by the users without having a specialized knowledge of it.
Simulation	The extent to which the technique is capable of dynamically simulating a business process.
Scope	The extent to which the process modelling elements are defined.

The comparative analysis of the business process modelling techniques is depicted in Table 2.1. Out of other techniques, Business Process Modelling

Notation is selected since it provides better readability and well-defined by the BPMN specification maintained by OMG group.

Table 2.1: Business Modeling Techniques (BPM) Comparison [10]

	Flexibility	Ease of Use	Understandability	Simulation	Scope
Flow Charts	• Simple to update • Easy to modify • Do not possess a sophisticated mechanism for modularising or packaging diagrams; hence invoking other processes from flow charts can be problematic	• Easy to learn • Easy to use • Limited set of symbols	• Easy to understand • Clear semantics of the constructs	• To build active flow charts, many commercial simulation tools are available that adopt flow charts as the underlying technique (e.g. iGrafx)	• Use as a technique to model processes • Does not have the means to explicitly represent services, events and rules
Petri Nets	• Graphical and precise mathematical notations • Suitable for the analysis and reengineering of business process models	• Small number of modelling elements to construct a model • Limited explicit expressivity – (non user-oriented technique) • difficult to adopt	• Require a certain level of expertise to model	• Use for transforming static process models into dynamic simulation models • Enables an inexperienced user to monitor how processes are executed	• The concepts of service, goal and role are not explicitly supported

	Flexibility	Ease of Use	Understandability	Simulation	Scope
Data Flow Diagrams	• Multiple levels of representation • Can modularise the representation of the process	• Small number of elements require to construct a model • The expressivity of the modelling elements facilitates the construction of a DFD model for inexperienced users	• Easy to understand • Easy to draw, improve and amend	• Not a technique that can easily support simulation	• Use four (4) basic elements for modelling business processes; process, data store, terminators and flow
Role Activity Diagrams (RAD)	• Processes can be refined and improved	• Set of symbols to describe processes • Easy to use	• Useful for large systems with many participants • Easy to understand	• Supports simulation by enabling detailed inspections of specific parts of the process	• Each role has attributes that govern its behaviour
Business Object Interaction diagrams	• Modularity and focused responsibility are available	• Expertise knowledge is required in mastering object-orientation when modelling business processes	• Knowledge of object-orientation is required	• Suitable to run simulations of business processes	• Sequence diagrams do not strictly support the concepts of process and activity

	Flexibility	Ease of Use	Understandability	Simulation	Scope
Business Process Modelling Notation (BPMN)	• Well-structured technique to model different types of processes • Allow the representation of extended models for each process • Enable flexible changes or improvement of any process in the extended model without affecting the original model	• Easy to use • Different kinds of flow of control and sequences • Complex diagramming technique	• All the users can understand the notations easily	• Support the construction of simulation models • Ability to test and visualize processes before implement	• Support all of the business process modelling elements
Business Use Cases	• These are textual descriptions of organisational processes; hence the narrative can be easily modified. • But this can be led to ambiguities and inconsistencies	• Textual narratives, hence can be learned quickly	• Business use cases are straight forward therefore, easy to understand	• Do not directly support simulation	• Support all of the business process modelling elements

2.3. Analysis of Modeling Tools & Plugins for BPMN

Once BPMN was selected as business process modelling technique for the study, a comparative analysis is done on the tools and plugins available for the BPMN. The Table 2.2 depicts the analysis and the criterion are derived from an analysis by R. Koncevičs, et al. [11]. As per the analysis 'Eclipse BPMN2.0 Modeler' plugin for the Eclipse IDE is selected as the modeling tool for this research.

Table 2.2: BPMN Modelling Tools & Plugins Comparison [11]

	Eclipse BPMN2.0	BizAgi	BonitaSoft	Oracle BPA suite	ProcessMaker	TIBCO Bus. Stud.	MS Visio	Visual Paradigm
Graphical user interface (GUI)	x	x	x	x	x	x	x	x
Supported notations								
BPMN	x	x	x	x		x	x	x
BPEL				x				
UML				x			x	x
Other				x			x	x
Publishing formats								
Image	x	x	x	x		x	x	x
Ms Word	x	x						
PDF	x	x	x				x	
Web	x	x	x	x		x		x
Export								
BPEL				x			x	
BPMN	x		x	x			x	x
XPDL			x	x			x	
XML	x			x		x	x	x
Import								
BPEL				x			x	
BPMN	x		x	x			x	x
XPDL				x			x	
XML				x		x	x	x
Open-Source	x				x			
Ability to implement user-defined reusable functionality	x	x	x	x	x	x	x	x
Change management of models	x	x	x	x	x		x	x
In-tool verification of notation	x	x	x	x		x	x	x
Ability to configure pre-defined objects in model	x	x	x	x	x	x	x	x

The chosen Eclipse BPMN 2.0 Modeler plugin is a free and open-source plugin and consists of a graphical user interface (GUI) and assist all the notations available in BPMN 2.0 specification. The plugin allows publishing diagrams as a MS Word, images (jpg, png, gif) or web content(html) and export & import formats such as BPMN and XML are supported. In-tool verification of BPMN notations is an added advantage.

2.4. Analysis of BPMN 2.0 Specification and Solidity Language

Upon selection of the tool for BPMN, an in-depth analysis of BPMN 2.0 specification is performed to determine the all existing business process modelling notations in BPMN.

Business process modelling diagrams being used to communicate to a broad audience. The Object Management Group (OMG) is publishing and maintaining BPMN specifications. BPMN version 2.0[2] is the latest specification and it provides an XML notation which describes a set of rules for the BPMN notations.

Solidity is a contract-oriented high-level programming language [12]. Latest version is 0.5.9[2] and documentation is available through their website. It is constructed to execute on-top of an Ethereum Virtual Machine (EVM).

[2] Documentation for the latest version of Solidity is available in https://solidity.readthedocs.io/en/v0.5.9/index.html

2.5. Existing Systems

2.6.1. Caterpillar

Caterpillar [13] [14] is an open-source business process execution engine for the Ethereum blockchain. Process modeling tool provides ability to formulate process models using BPMN process elements. Also consists of an execution panel to create new instances and monitor state of the process model.

Caterpillar specializes in work-flows of the process models and provides a rich set of restful APIs to create, view, update and remove process models. Diagram elements provides an abstract view of the process model and lower details such as variable declarations require annotating the diagram elements [13]. Once process model is created, execution panel provides ability to create new process instances on the blockchain and state or hand-offs between multiple parties are visually represented in the execution panel. Further all the functionalities of the execution panel exposed through the APIs as well.

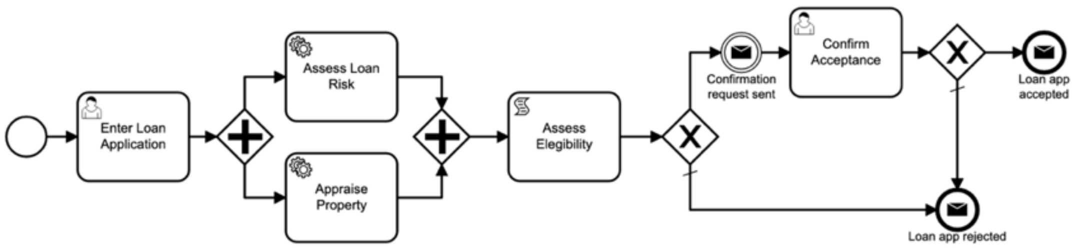


Figure 2.2: A Process Model Example in Caterpillar [13]

2.6.2. Unibright

Unibright [15] is a commercial unified framework for blockchain based integrations. Unibright provides a template based visual experience for the process integration. Automated smart contract generation can be done for the diagrams and workflows created using Business Workflow Designer component. Secondly, generated code can be deployed into blockchain using 'Contract Lifecycle Manager' component which is responsible in creating, viewing, updating and deleting contracts.

Most of the development information are not accessible to the public hence it is a commercial solution. However, only a public demo is available in unibright.io[3] website. As per the demo, only a multi-party approval template is available with a limited set of process elements (only 'approver' and 'feedback'). A multi-party approval example is depicted in Figure 2.3. However, the support for the BPMN notations cannot be verified with the limited information available.

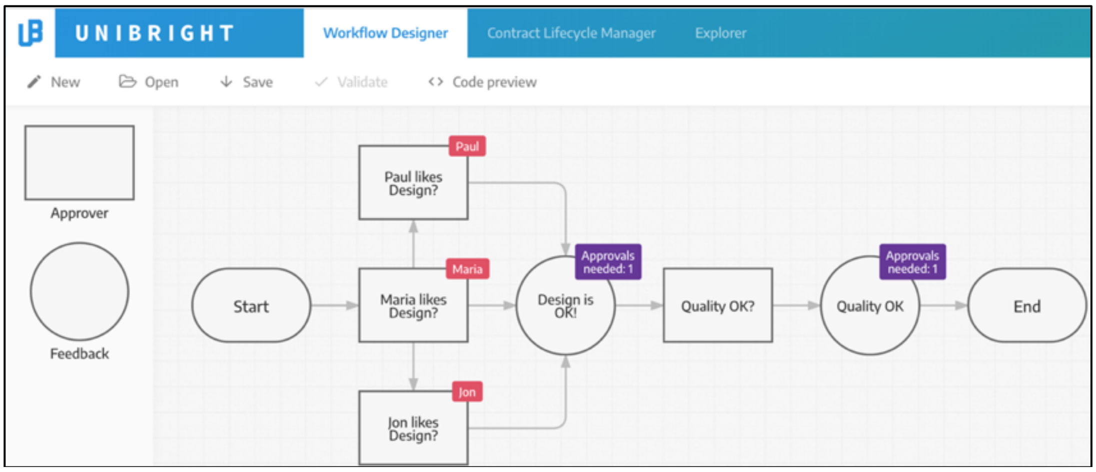


Figure 2.3: Multiparty Approval Example using Unibright

2.6.3. Petri-Nets Translation and Reduction Approach

García-Bañuelos et. al [16] proposes a Petri-Nets translation and reduction based approach that make use of Petri-Nets as an intermediate representation when converting BPMN process models into Solidity language. This process has several reduction phases using data conditions as explained in [16]. Figure 2.4 explains the reduction and translation process of a BPMN diagram with this approach.

As per the authors, gas consumption is minimized by a special process that encodes current state of the process in a space optimized bit-array data structure. Further, overhead is minimized by the use of "factory" and "instances" pattern for deploying multiple instances of the same process model. These changes are done on top of its previous versions and the results has shown significant improvement in gas consumption.

[3] A demo example of the Unibright is available in https://authentication.unibright.io/ online, accessed 18-01-2020.

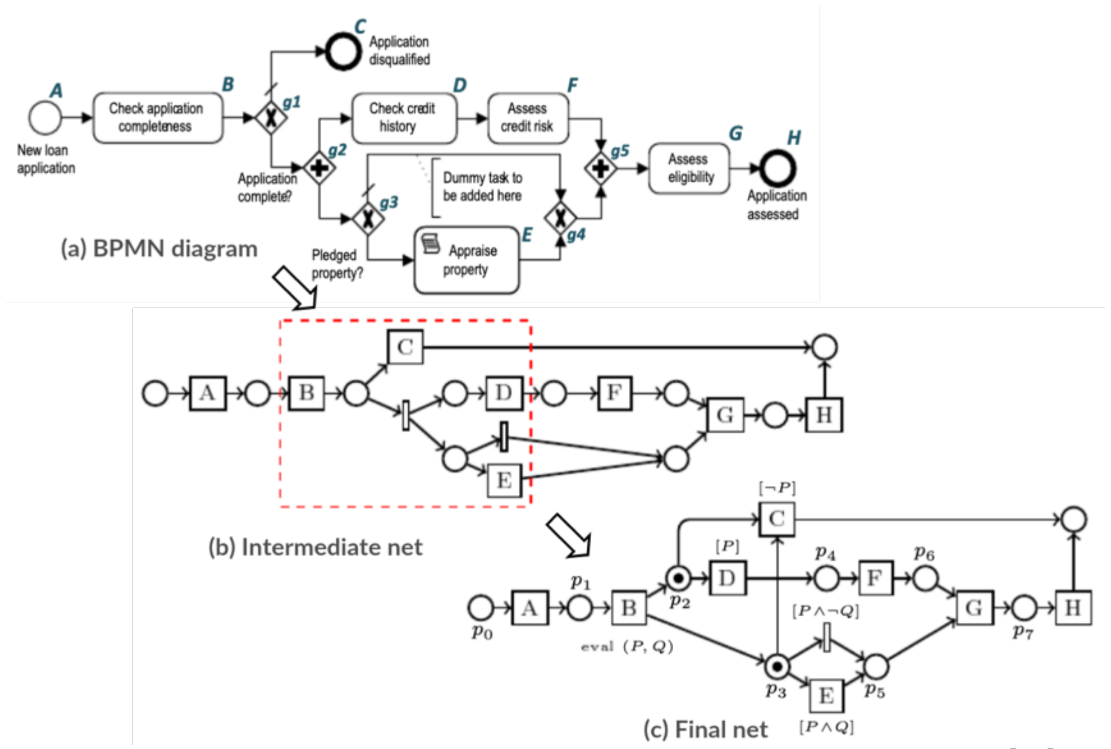


Figure 2.4: An example of PetriNets Translation and Reduction Approach [16]

2.6.4. Choreography Diagrams Method

Another method is to use Choreography diagrams [17] defined in BPMN 2.0 to create process models for the blockchains. Figure 2.5 depicts a sample process model using choreography diagrams for a supply chain example.

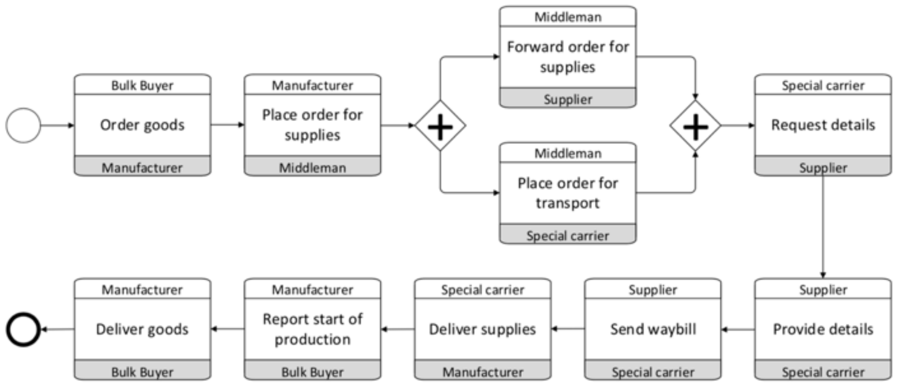


Figure 2.5: A Supply Chain Example using Choreography Based Approach [17]

In order to follow up of state of the process instance, a sequence of storage variables is used. In the code generation algorithm, BPMN Tasks and AND-join gateways are mapped into functions. The first task is triggered by 'Init( )' function and other tasks are invoked by the triggers defined as

‘Task$_i$( )’. function ‘JoinGateway$_i$( )’ is invoked internally whenever controlling the process flow is required. These triggers are being executed in runtime with the API calls from the C-Monitor (Choreography Monitor); a web-based process execution panel [18].

2.6.5. Extended Choreography Diagrams Approach

Ladleif et al. [7], suggests a backward compatible extended version of BPMN choreography diagrams to generate the Solidity smart contracts. As per the Figure 2.6; an interaction between an initiated and a respondent is represented as a choreography task (marked in rounded rectangles). An interaction is depicted in shaded background.

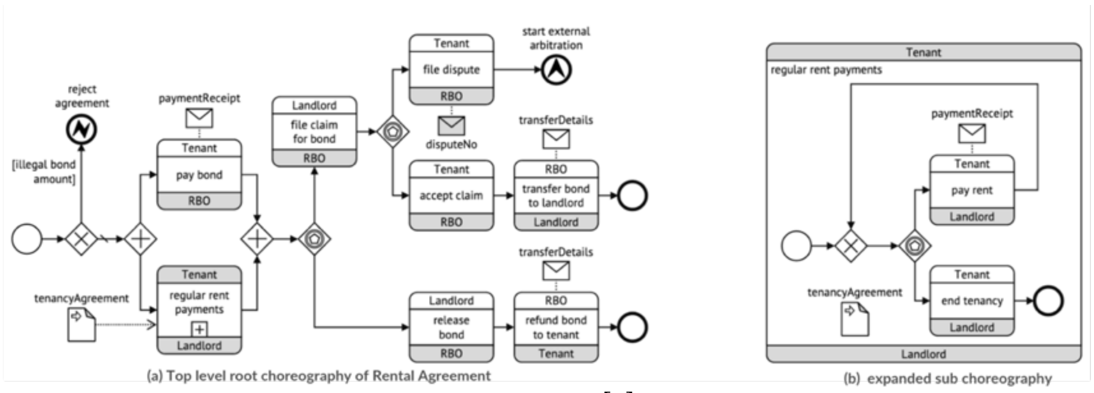


Figure 2.6: A Rental Agreement Example [7] using Extended Choreography Diagrams

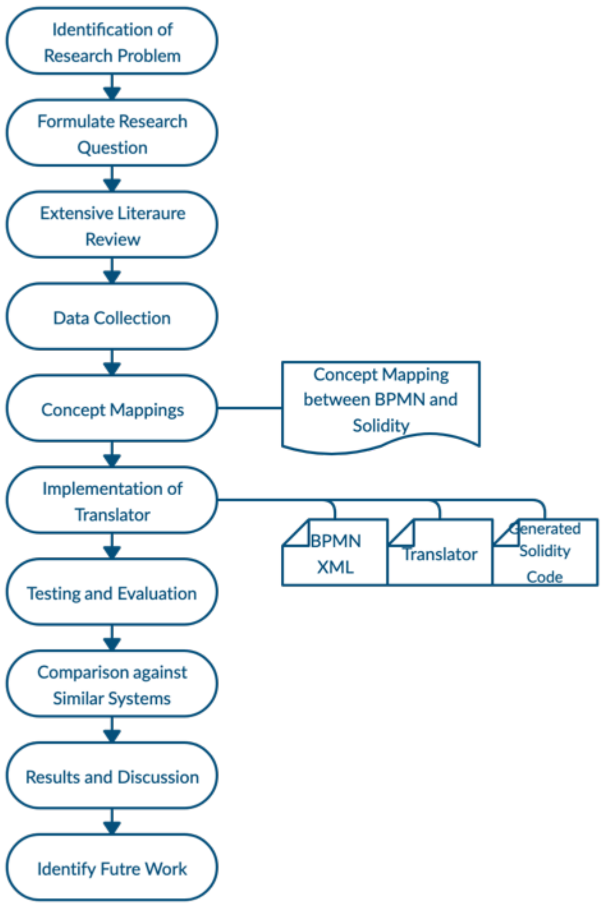
An envelope depicts an initiating message and Shaded envelope denotes the response. Authors claims that there are two main limiting factors in choreography diagrams defined in BPMN specification, (1) ownership that enforces which elements (2) observability that enforces who knows what. Another major limitation is the absence of data structures which are already available in other sub models [7]. Due to these limitations, authors are proposing a backward compatible extension for the BPMN choreography diagrams.

3. RESEARCH METHODOLOGY

The literature review chapter provided a theoretical base exploring various business process modelling techniques, process modelling tools & plugins and evaluating the state-of-the-art related work. Research methodology provides clear step-by-step road to solve the research problem. In this chapter, a suitable research methodology will be proposed in order to achieve proposed goals and objectives.

3.1. High Level Steps of the Research Methodology

Below Figure 3.1 depicts the high-level steps and phases followed for the research.



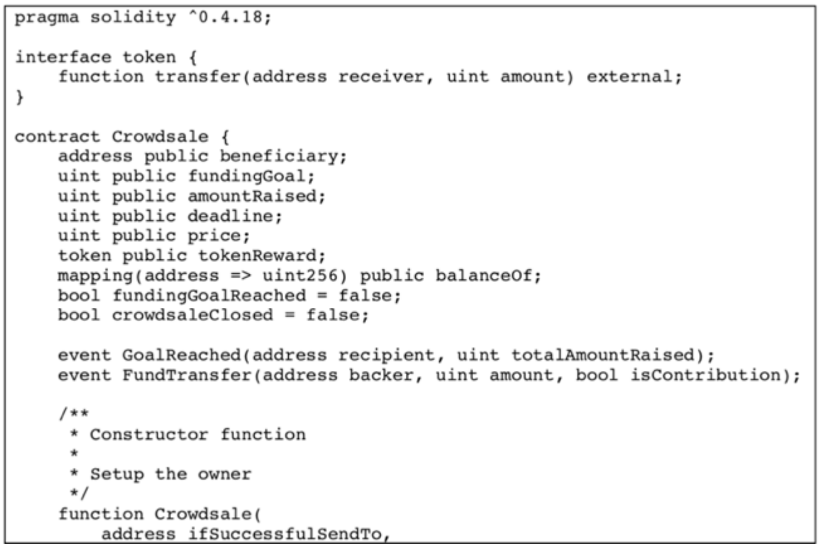
Identification of Research Problem
Formulate Research Question
Extensive Literaure Review
Data Collection
Concept Mappings
Concept Mapping between BPMN and Solidity
Implementation of Translator
BPMN XML
Translator
Generated Solidity Code
Testing and Evaluation
Comparison against Similar Systems
Results and Discussion
Identify Futre Work

Figure 3.1: High level view of the Research Methodology

From the '3.2: Data Collection' phase, all the pending steps have been discussed in great details.

3.2. Data Collection

In the data collection phase, mainly two types of data have been collected, (1) Smart contract implementations in Solidity code and (2) business contracts. For the smart contract implementations, online repositories such as GitHub, GitLab and blog articles such as Medium have been analyzed. More than sixty (60) contracts from various domains such as insurance, banking & finance, shipping & logistics were collected. Initial filtering was done using the ability to deploy the contracts in the blockchain. In Figure 11, a Solidity smart contract example called 'Crowdsale' is depicted. Crowdsale allows collecting of capital for a new business venture in small amounts from a large number of participants.



```
pragma solidity ^0.4.18;

interface token {
    function transfer(address receiver, uint amount) external;
}

contract Crowdsale {
    address public beneficiary;
    uint public fundingGoal;
    uint public amountRaised;
    uint public deadline;
    uint public price;
    token public tokenReward;
    mapping(address => uint256) public balanceOf;
    bool fundingGoalReached = false;
    bool crowdsaleClosed = false;

    event GoalReached(address recipient, uint totalAmountRaised);
    event FundTransfer(address backer, uint amount, bool isContribution);

    /**
     * Constructor function
     *
     * Setup the owner
     */
    function Crowdsale(
        address ifSuccessfulSendTo,
```

Figure 3.2: Crowdsale Solidity Smart Contract[4]

Further, more than twenty (20) smart contracts which covers business scenarios such as e-voting, loan-process, rental-payments, crowd-funding…etc. have been further analysed for the business use case. As the first steps, sequence diagrams are used to derive the business ue case. Figure 3.3 depicts the sequence diagram for the same Crowdsale example in Figure 3.2.

[4] Solidity smart contract implementation for Crowdsale is available in https://github.com/ethereum/ethereum-org/blob/983ce0b84dbb4008bbd78c9cf5d0e563a619aaae/dist/crowdsale.html online, accessed 19-11-2019

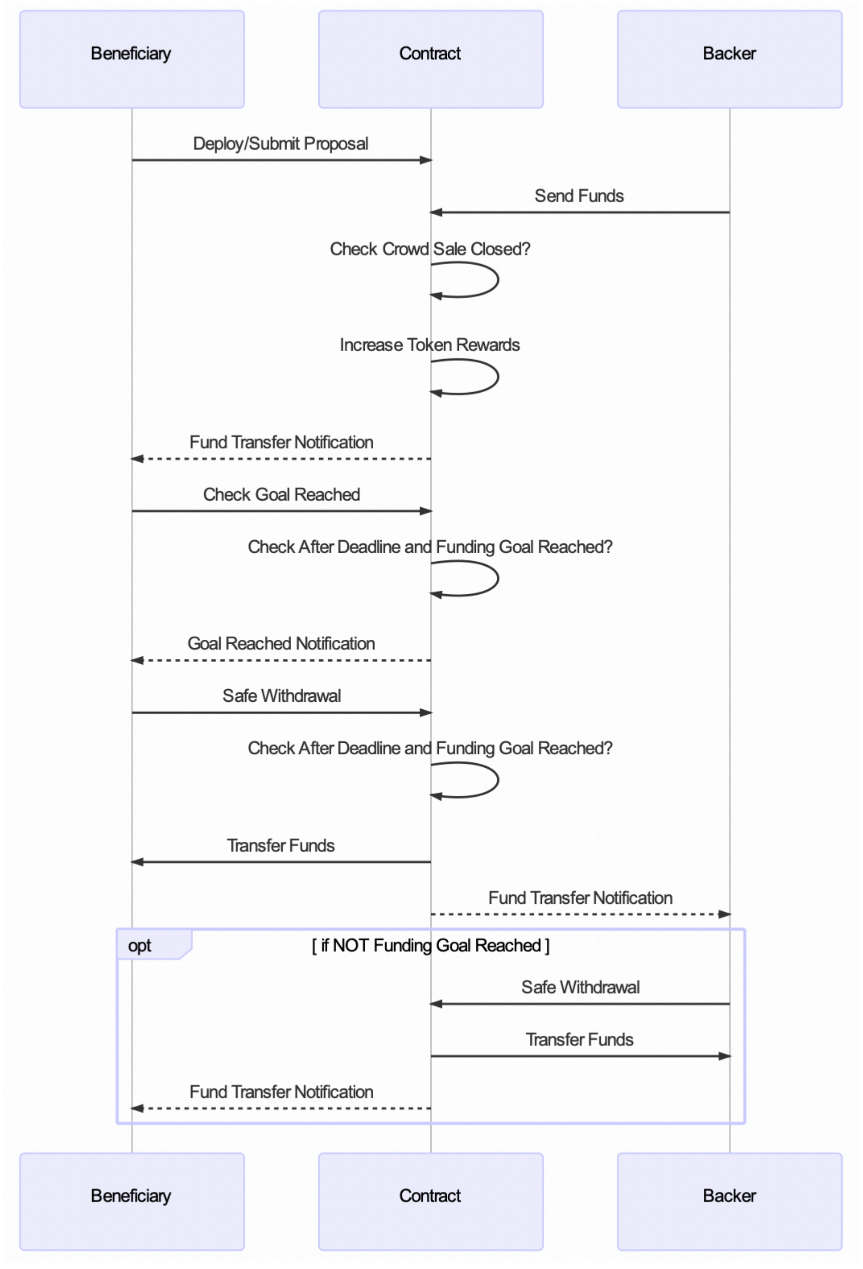

Beneficiary
Contract
Backer
Deploy/Submit Proposal
Send Funds
Check Crowd Sale Closed?
Increase Token Rewards
Fund Transfer Notification
Check Goal Reached
Check After Deadline and Funding Goal Reached?
Goal Reached Notification
Safe Withdrawal
Check After Deadline and Funding Goal Reached?
Transfer Funds
Fund Transfer Notification
opt
[if NOT Funding Goal Reached]
Safe Withdrawal
Transfer Funds
Fund Transfer Notification
Beneficiary
Contract
Backer

Figure 3.3: Sequence Diagram Representation of Crowd Sale Example

3.3. Implementation of the Translator

The outcome of the concept mapping between BPMN and Solidity in section 3.4 (page 36) is the main input for the development of this translator. The two key components of implementation of this translator are listed as below;

(1) BPMN Modeler Integration
(2) BPMN to Solidity Translation

3.3.1. BPMN Modeler Integration

The topmost layer of the translator. In theory, BPMN Modeler can be any tool that can support BPMN 2.0 specification. Facilitating a graphical user interface (GUI) to manipulate the BPMN process elements of the process model is the main functionality of this layer. In addition, XSD verification against the BPMN specification provides instant error reporting to the user to alter the process model whenever needed.

As per analysis of BPMN modelling tools & plugins in section 2.3 (page 22), Eclipse BPMN 2.0 Modeler plugin[5] is selected as the modelling tool for this purpose. Further this allows most advanced features such as 'I/O Specification of Tasks' to define input and output for a particular task.

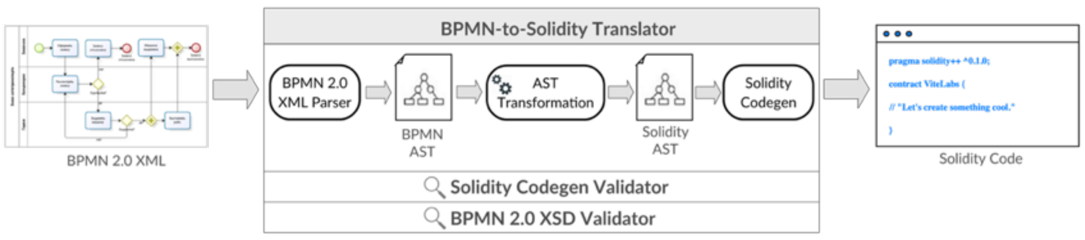
3.3.2. BPMN to Solidity Translation

This layer is responsible for the BPMN 2.0 Abstract Syntax Tree (AST) validation and Solidity smart contract code generation. Validation of the process model is done is two phases. Firstly, using XSD validation of the BPMN XML provided by OMG group. Secondly, using AST validation consists of a set of custom rules defined before and after AST generation.

High level steps followed to implement BPMN to Solidity translator is listed below;

Step I:	ANTLRv4 Parser Generation for BPMN XML
Step II:	Implement XML Schema Definition Validator for BPMN
Step III:	BPMN Abstract Syntax Tree (AST) Generation
Step IV:	Solidity Abstract Syntax Tree (AST) Generation
Step V:	Solidity Code Generation

[5] BPMN2 Modeler Plugin is a free and open source BPMN plugin for Eclipse IDE and it is available in https://www.eclipse.org/bpmn2-modeler online, accessed 02-03-2020



```
pragma solidity++ ^0.1.0;
contract ViteLabs {
// "Let's create something cool."
}
```

Figure 3.4: Top Level View of BPMN to Solidity Translator

Step I: ANLRv4 Parser Generation for BPMN XML

Once the user model the process model using BPMN modelling tool it can be exported into a BPMN XML file named as *"<filename>.bpmn"*. This XML file contains the meta data of all BPMN elements used and their relationships between each other.

First step of generating BPMN AST is to parse the BPMN XML. A BPMN XML is structured with two main parts;

1. **Process Elements**: Definitions of the BPMN nodes and their properties and relationships.
2. **BPMN Diagram Elements**: Contains all graphical information of the BPMN nodes such as their location, colours…etc.

For this research, we are only interested in 'Process Elements' to generate the BPMN AST and 'BPMN Diagram Elements' are simply ignored. BPMN ANTLR g4 grammar files are generated with adding few modifications for the already existing XML ANTLR g4 grammar file[6] which is available online. The BPMN grammar and lexer files are depicted in
Figure 3.6 and Figure 3.7.

It is important to point out that the parser generation step is one-time task (unless there's modification to the grammar files) and translation events are being handled by the BPMN parser listener implementation for the generated parser.

[6] The sample ANTLRv4 XML grammar file used is available in https://github.com/antlr/grammars-v4/blob/master/xml/XMLParser.g4 online, accessed 13-11-2019

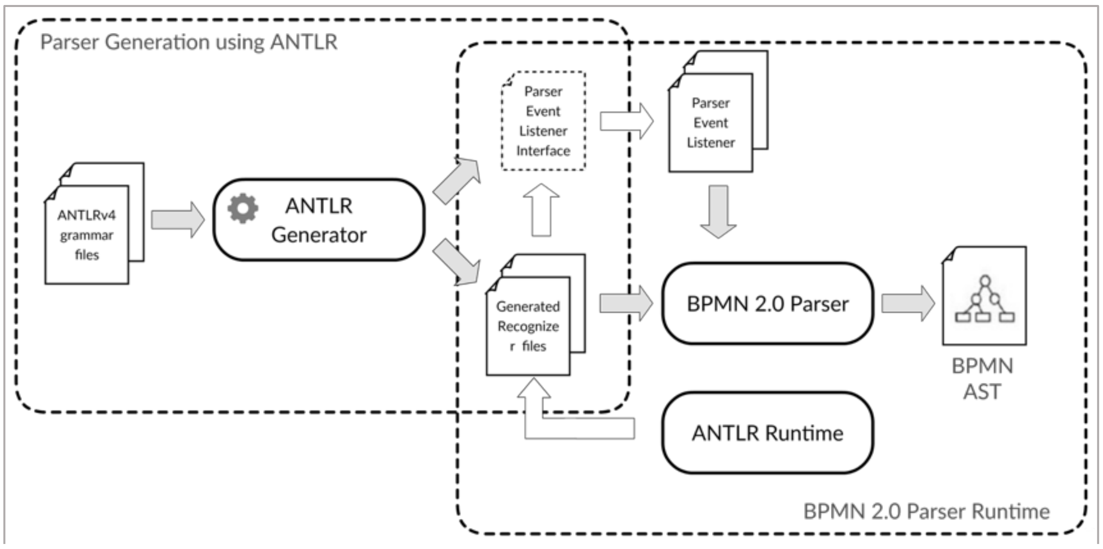
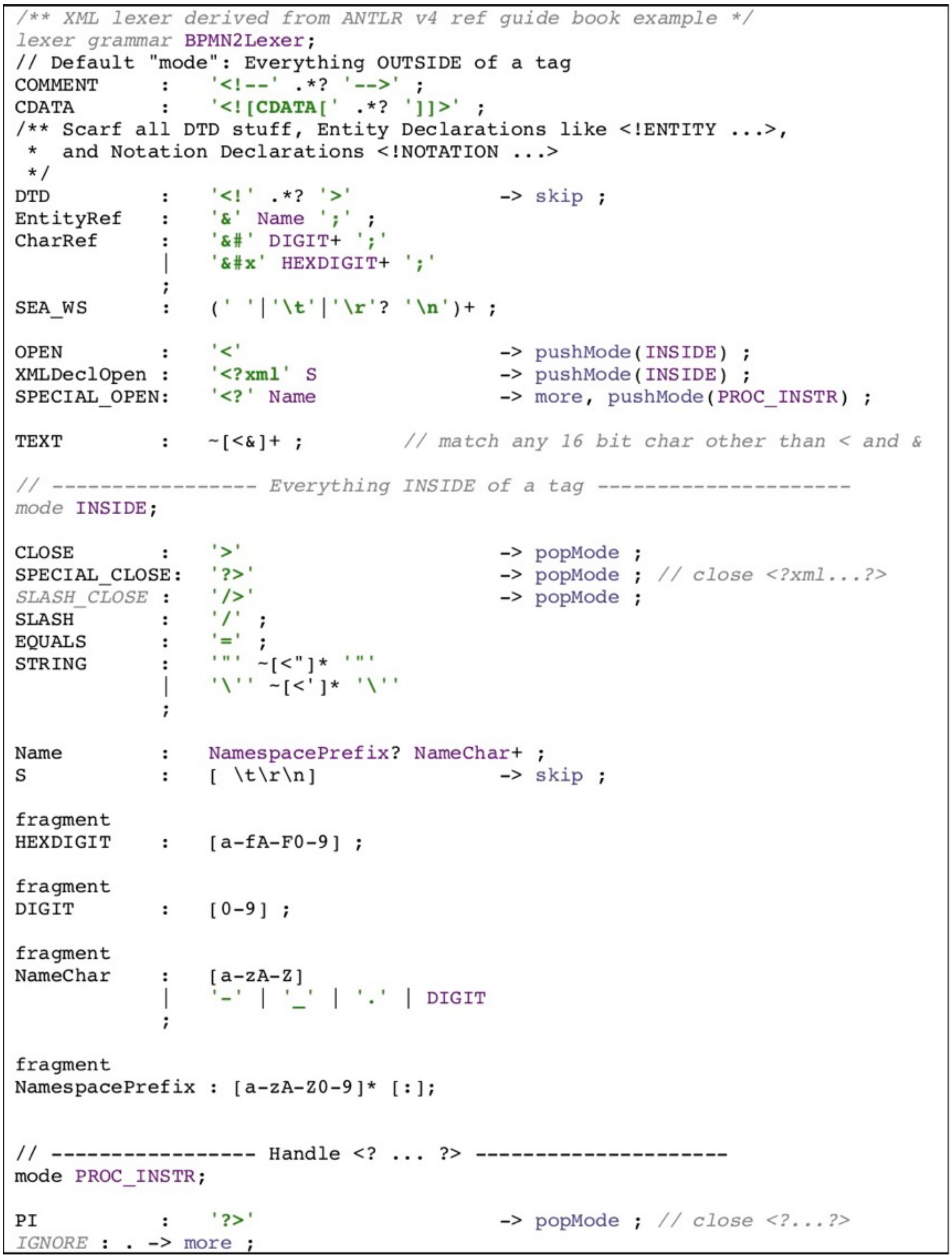


Figure 3.5: ANLRv4 Parser Generation for BPMN XML

```
parser grammar BPMN2;

options { tokenVocab=BPMN2Lexer; }

document              :   prolog? misc* element misc*;

prolog                :   XMLDeclOpen attribute* SPECIAL_CLOSE ;

content               :   chardata?
                          ((element | reference | CDATA | PI |
COMMENT) chardata?)* ;

selfClosingElement    : '<' Name attribute* '/>';
blockElement          : '<' Name attribute* '>' content '<' '/' Name
'>';

element               :   selfClosingElement
                      |   blockElement
                      ;

reference             :   EntityRef | CharRef ;

attribute             :   Name '=' STRING ; // Our STRING is
AttValue in spec

/** ``All text that is not markup constitutes the character data
of
 *  the document.''
 */
chardata              :   TEXT | SEA_WS ;

misc                  :   COMMENT | PI | SEA_WS ;
```

Figure 3.6: BPMN ANTLR g4 Grammar



```
/** XML lexer derived from ANTLR v4 ref guide book example */
lexer grammar BPMN2Lexer;
// Default "mode": Everything OUTSIDE of a tag
COMMENT     :   '<!--' .*? '-->' ;
CDATA       :   '<![CDATA[' .*? ']]>' ;
/** Scarf all DTD stuff, Entity Declarations like <!ENTITY ...>,
 *  and Notation Declarations <!NOTATION ...>
 */
DTD         :   '<!' .*? '>'              -> skip ;
EntityRef   :   '&' Name ';' ;
CharRef     :   '&#' DIGIT+ ';'
            |   '&#x' HEXDIGIT+ ';'
            ;
SEA_WS      :   (' '|'\t'|'\r'? '\n')+ ;

OPEN        :   '<'                       -> pushMode(INSIDE) ;
XMLDeclOpen :   '<?xml' S                 -> pushMode(INSIDE) ;
SPECIAL_OPEN:   '<?' Name                 -> more, pushMode(PROC_INSTR) ;

TEXT        :   ~[<&]+ ;         // match any 16 bit char other than < and &

// ----------------- Everything INSIDE of a tag ---------------------
mode INSIDE;

CLOSE       :   '>'                       -> popMode ;
SPECIAL_CLOSE:  '?>'                      -> popMode ; // close <?xml...?>
SLASH_CLOSE :   '/>'                      -> popMode ;
SLASH       :   '/' ;
EQUALS      :   '=' ;
STRING      :   '"' ~[<"]* '"'
            |   '\'' ~[<']* '\''
            ;

Name        :   NamespacePrefix? NameChar+ ;
S           :   [ \t\r\n]                 -> skip ;

fragment
HEXDIGIT    :   [a-fA-F0-9] ;

fragment
DIGIT       :   [0-9] ;

fragment
NameChar    :   [a-zA-Z]
            |   '-' | '_' | '.' | DIGIT
            ;

fragment
NamespacePrefix : [a-zA-Z0-9]* [:];


// ----------------- Handle <? ... ?> ---------------------
mode PROC_INSTR;

PI          :   '?>'                      -> popMode ; // close <?...?>
IGNORE : . -> more ;
```

Figure 3.7: BPMN ANTLR g4 Lexer

Step II: Implement XML Schema Definition Validator for BPMN

An accumulated version of the BPMN XSDs available online[7] are being used for the validation of the input BPMN XML files. This step makes sure that the input process model is verified and valid for the further steps.

Step III: BPMN Abstract Syntax Tree (AST) Generation

Whenever parsing an input file, 'BPMN Parser Listener' will receive events from the BPMN parser. The listener builds the BPMN AST with the events it receives. The BPMN Abstract Syntax Tree (AST) contains information such as node type, node name, attributes and child nodes.

Step IV: Solidity Abstract Syntax Tree (AST) Generation

In order to generate Solidity Abstract Syntax Tree (AST), BPMN AST is used as an input. This AST transformation is driven by the BPMN to Solidity concept mapping formulated in section 3.4(page 36). Secondary validation of the translator is executed during this phase.

Step V: Solidity Code Generation

As the last step of the translator, Solidity code is generated using a tree visitor named as 'Solidity Codegen Tree Visitor' on top of the Solidity Abstract Syntax Tree (AST).

[7] A set of BPMN XML Schema Definitions are available in https://www.omg.org/spec/BPMN/2.0/About-BPMN/ online, accessed 21-12-2019

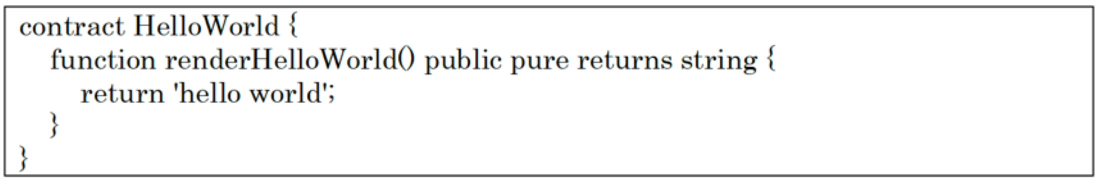
3.4. BPMN-to-Solidity Abstract Syntax Tree (AST) Transformation

Abstract Syntax Tree transformation from BPMN AST to Solidity AST is elaborated in this section. The concepts (or XML nodes) in BPMN XML file are being mapped from BPMN to Solidity smart contract language constructs. The outcome of this transformation rules set is being used as an input for translator implementation. In brief, AST transformation covered in three key aspects,

1. BPMN Sub-Models to Solidity Mapping
2. BPMN Elements to Solidity Mapping
3. BPMN Representation of Solidity Top Level Constructs

<u>**Example: Hello World**</u>

Hello World example exhibits how a simple process model in BPMN is being translated into a Solidity smart contract. For the simplicity, let's assume that the expected code (refer Figure 3.8) of the contract is known in advance and it needs to return a 'hello world' text phrase upon a trigger. It is noteworthy that the source BPMN XML depicted in Figure 3.11 is the generated from the BPMN process modeling tool with the process model defined in BPMN(refer Figure 3.9).



```
contract HelloWorld {
    function renderHelloWorld() public pure returns string {
        return 'hello world';
    }
}
```

Figure 3.8: Hello-World Example

<u>**Example: Hello World – BPMN Process Model**</u>

A BPMN process model is created using process modelling tool as the initial step (refer Figure 3.9). The BPMN process elements used for this example are namely a Pool (or Participant), a Start Event, a Task and an End Event. Additionally, need to configure a I/O output for the Task to represent the return value (refer Figure 3.10).

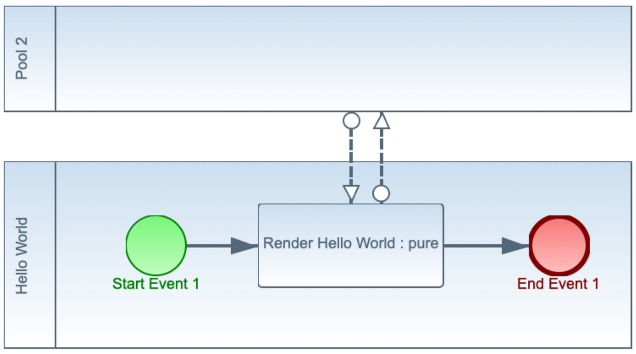
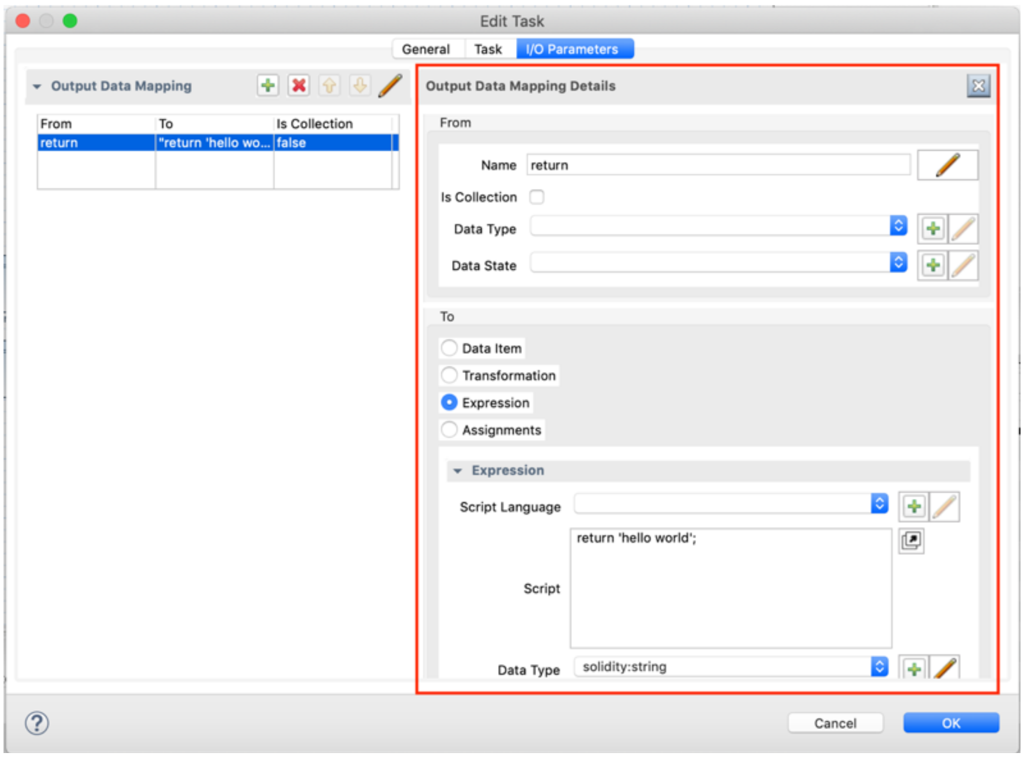


Figure 3.9: BPMN Process Model for the Hello-World Example



Output Data Mapping

From	To	Is Collection
return	"return 'hello wo...	false

Output Data Mapping Details

From

Name: return

Is Collection: ☐

Data Type:

Data State:

To

○ Data Item
○ Transformation
● Expression
○ Assignments

Expression

Script Language:

Script: return 'hello world';

Data Type: solidity:string

Figure 3.10: Configure A Return Text for the Hello-World Task

<u>Example: Hello World – BPMN XML</u>

After completing the process model, generated XML can be retrieved in source view of the modelling tool. For instance, BPMN 2.0 Modeler provides switching source view on-top of the process model diagram. For the above 'Hello World' example, generated XML is listed in Figure 3.11.

```
<?xml version="1.0" encoding="UTF-8"?>
<!-- origin at X=0.0 Y=0.0 -->
<bpmn2:definitions xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xmlns:bpmn2="http://www.omg.org/spec/BPMN/20100524/MODEL"
xmlns:bpmndi="http://www.omg.org/spec/BPMN/20100524/DI"
xmlns:dc="http://www.omg.org/spec/DD/20100524/DC"
xmlns:di="http://www.omg.org/spec/DD/20100524/DI"
xmlns:xs="http://www.w3.org/2001/XMLSchema" id="Definitions_1"
exporter="org.eclipse.bpmn2.modeler.core" exporterVersion="1.4.3.Final-
```

```
v20180418-1358-B1">
  <bpmn2:itemDefinition id="ItemDefinition_1090" isCollection="false"
structureRef="solidity:string"/>
  <bpmn2:collaboration id="Collaboration_1" name="Collaboration 1">
    <bpmn2:participant id="Participant_1" name="Hello World"
processRef="Process_1"/>
    <bpmn2:participant id="Participant_2" name="Pool 2"/>
    <bpmn2:messageFlow id="MessageFlow_1" sourceRef="Participant_2"
targetRef="Task_2"/>
    <bpmn2:messageFlow id="MessageFlow_2" sourceRef="Task_2"
targetRef="Participant_2"/>
  </bpmn2:collaboration>
  <bpmn2:process id="Process_1" name="Hello World Process"
definitionalCollaborationRef="Collaboration_1" isExecutable="false">
    <bpmn2:startEvent id="StartEvent_1" name="Start Event 1">
      <bpmn2:outgoing>SequenceFlow_1</bpmn2:outgoing>
    </bpmn2:startEvent>
    <bpmn2:endEvent id="EndEvent_1" name="End Event 1">
      <bpmn2:incoming>SequenceFlow_2</bpmn2:incoming>
    </bpmn2:endEvent>
    <bpmn2:sequenceFlow id="SequenceFlow_1" sourceRef="StartEvent_1"
targetRef="Task_2"/>
    <bpmn2:sequenceFlow id="SequenceFlow_2" sourceRef="Task_2"
targetRef="EndEvent_1"/>
    <bpmn2:task id="Task_2" name="Render Hello World : pure">
      <bpmn2:incoming>SequenceFlow_1</bpmn2:incoming>
      <bpmn2:outgoing>SequenceFlow_2</bpmn2:outgoing>
      <bpmn2:ioSpecification id="InputOutputSpecification_7">
        <bpmn2:dataOutput id="DataOutput_1" name="return"/>
        <bpmn2:inputSet id="InputSet_7" name="Input Set 7"/>
        <bpmn2:outputSet id="OutputSet_7" name="Output Set 7">
          <bpmn2:dataOutputRefs>DataOutput_1</bpmn2:dataOutputRefs>
        </bpmn2:outputSet>
      </bpmn2:ioSpecification>
      <bpmn2:dataOutputAssociation id="DataOutputAssociation_1">
        <bpmn2:sourceRef>DataOutput_1</bpmn2:sourceRef>
        <bpmn2:assignment id="Assignment_1">
          <bpmn2:from xsi:type="bpmn2:tFormalExpression"
id="FormalExpression_1">DataOutput_1</bpmn2:from>
          <bpmn2:to xsi:type="bpmn2:tFormalExpression"
id="FormalExpression_3" evaluatesToTypeRef="ItemDefinition_1090">return
'hello world';</bpmn2:to>
        </bpmn2:assignment>
      </bpmn2:dataOutputAssociation>
    </bpmn2:task>
  </bpmn2:process>
  <bpmndi:BPMNDiagram id="BPMNDiagram_1">
    <!—Omitted for the simplicity-->
  </bpmndi:BPMNDiagram>
</bpmn2:definitions>
```

Figure 3.11: An Excerpt of Hello-World Example's BPMN XML

In order to generate expected Solidity smart contract code in Figure 3.8, Translator first parses the BPMN XML file in Figure 3.11. Depicted in hen the BPMN AST is generated as Figure 3.12. The outcome of the AST Transformation is the Solidity AST which is depicted in Figure 3.13.

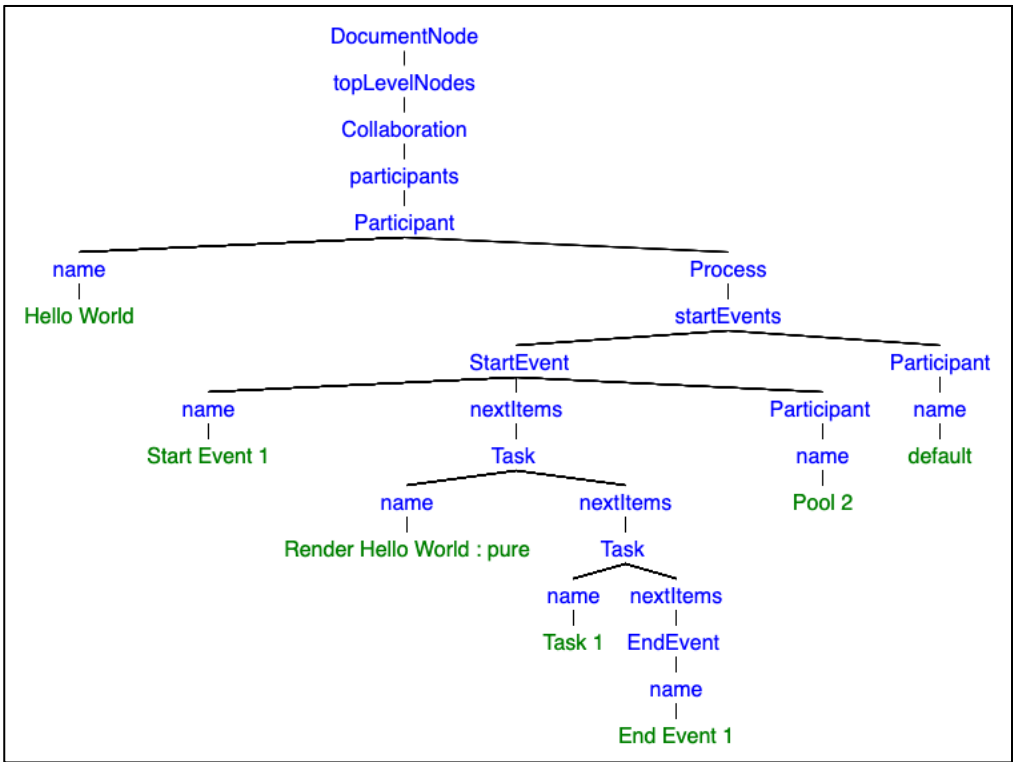


Figure 3.12: HelloWorld example's BPMN AST

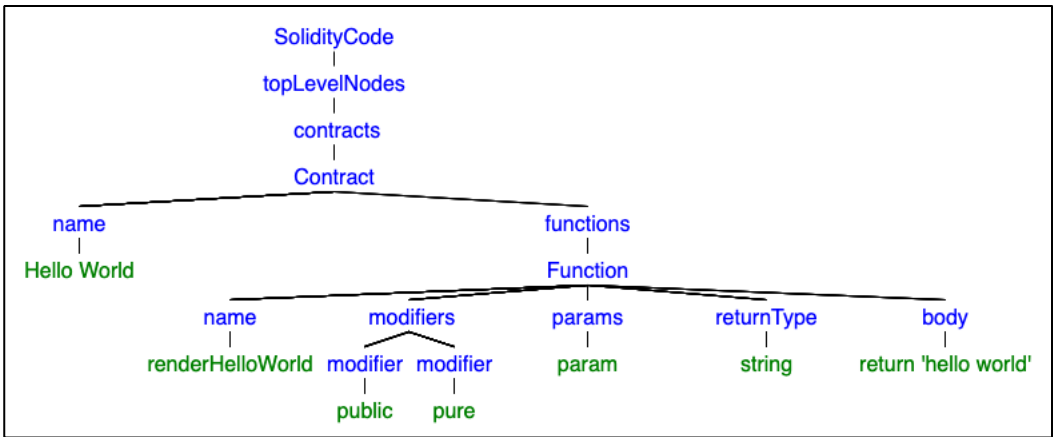


Figure 3.13: HelloWorld example's Solidity AST

Below section explains how BPMN XML nodes are being mapped into Solidity language constructs in great details.

3.4.1. BPMN Sub-Models to Solidity Mapping

Sub models are the diagram types available in BPMN specification. Three (3) main sub-models are defined in BPMN 2.0 specification as listed below;

1. **Processes (Orchestration)**

   Processes are the sequence of activities in an organization with the objective of carrying out a work. By default, processes are private. Whenever a process communicates with another process or participant, the process becomes a public process.

   It is worth noting that this 'public' and 'private' process concepts are being used with same semantic meaning when translating into functions of the Solidity smart contract code.

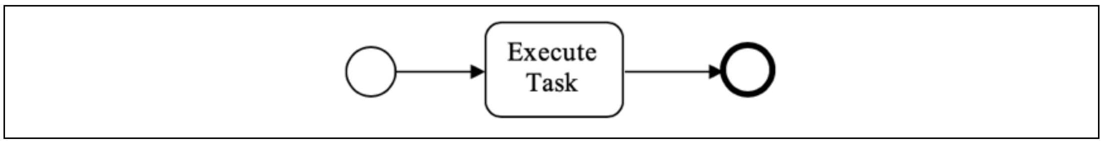


Figure 3.14: BPMN private process example

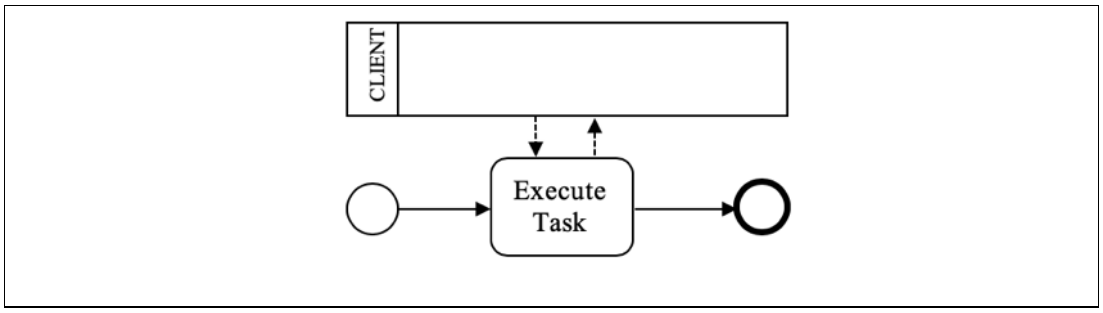


Figure 3.15: BPMN public process example

2. **Choreographies**

   Choreographies defined a different set of diagram elements in BPMN specification. Usually these diagrams are existing between interacting participants.

   Since it more focused in communication between multiple parties and non-existence of data structures such as data objects in combination with the scope and time limitations of the study, these types of diagrams are not supported by the translator.

3. **Collaborations**

   Collaborations are the combination of participants and processes and other diagram types. A collaboration contains two or more participants exchanging messages between each other.

When mapping the process model into Solidity language constructs, the name of the pool (or participant) become the name of the contract. For instance, in Figure 3.16 (page 41), the pool name 'Hello World' will generate a smart contract with the name 'hello World'. Since other dummy pool has no BPMN elements it won't create any smart contracts. Further, any task available will be mapped into functions. Any task which interacts with an external participant will become public, otherwise it will be private.

3.4.2. BPMN Elements to Solidity Mapping

BPMN elements in the specification are listed below in the order of importance;

(i) BPMN Events

1. BPMN Start Event and End Event

As depicted in Figure 3.16, the boundary line of the function is defined between a 'Start Event' and an 'End Event'. There can be multiple 'Start Events' but only single triggered event for a 'Participant'. Further, there can be multiple 'End Events' but only single 'End event' for a given request flow of a 'Participant'.

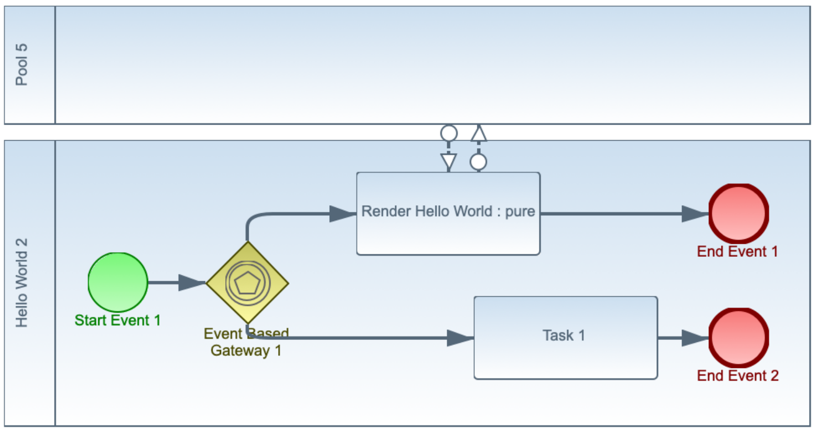


Figure 3.16: An Example for Start Events and End Events

Above process model defined in BPMN (Figure 3.16) will generate the below Solidity code depicted in

```
contract HelloWorld2 {
   function renderHelloWorld() public pure returns string {
      return 'hello world';
   }
   function task1() {
   }
}
```

Figure 3.17.

```
contract HelloWorld2 {
   function renderHelloWorld() public pure returns string {
      return 'hello world';
   }
   function task1() {
   }
}
```

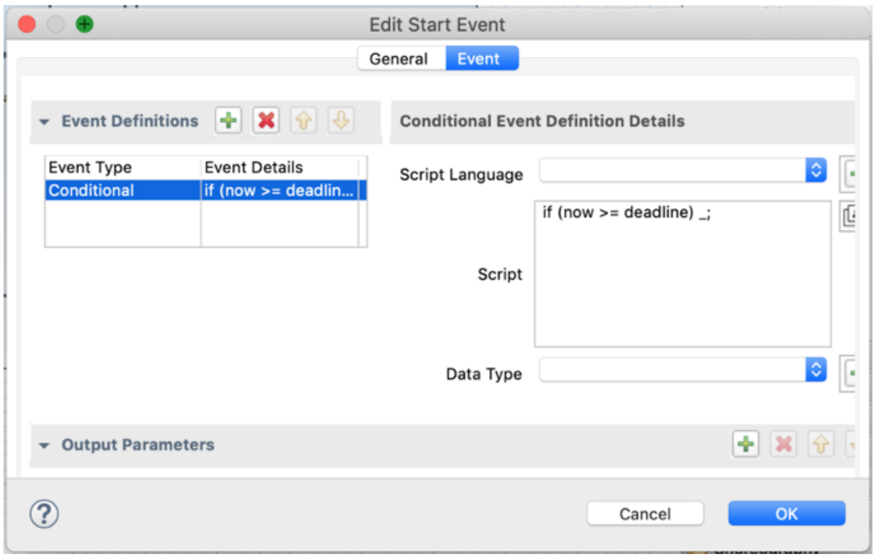
<u>**Generated Solidity Code**</u>

```
contract HelloWorld2 {
   function renderHelloWorld() public pure returns string {
      return 'hello world';
   }
   function task1() {
   }
}
```

Figure 3.17: Solidity Code for Start Event and End Event Example

2. BPMN Conditional Start Event

The use of these types of Start Event is a common pattern in BPMN diagrams. These 'Conditional Start Events' will be mapped into user defined 'modifiers' in Solidity language. Further, All the connecting 'Tasks' will contain the inline modifier name in the function signature and the implementation of the modifier will be added as a top-level construct to the smart contract implementation.



Event Type	Event Details
Conditional	if (now >= deadlin...

Script Language:

Script: if (now >= deadline) _;

Data Type:

Figure 3.18: Configuring Conditional Start Event

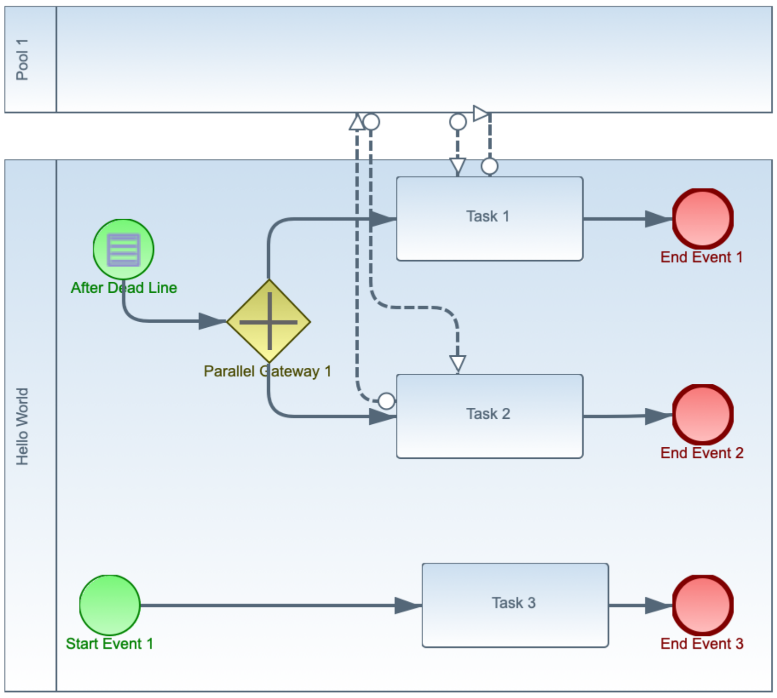


Figure 3.19: An Example for the Conditional Start Events

Above process model defined in BPMN (Figure 3.19) will generate the below Solidity code depicted in Figure 3.20;

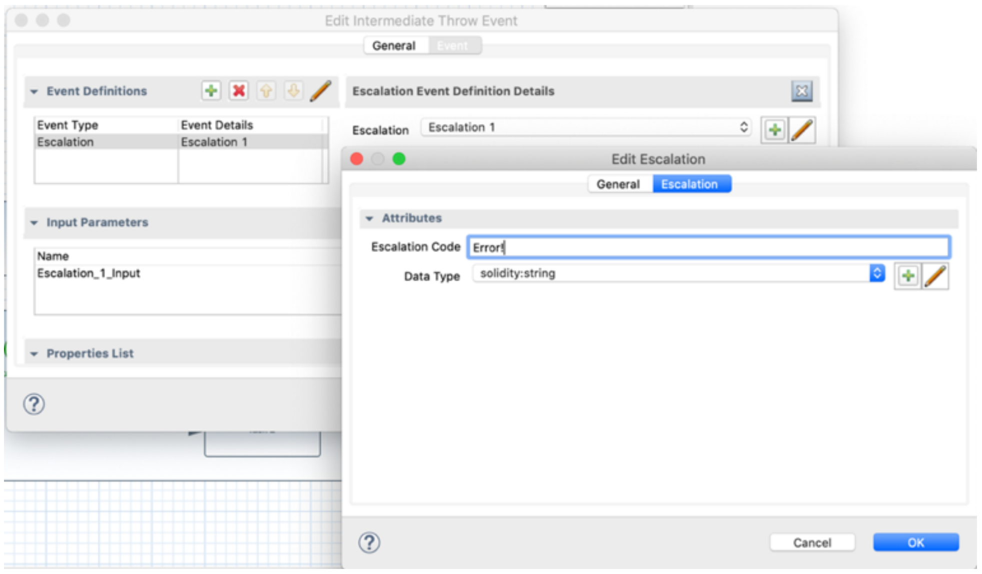
Generated Solidity Code

```
contract HelloWorld {
   function task1() public afterDeadLine {
   }

   function task2() public afterDeadLine {
   }

   function task3() {
   }

   modifier afterDeadLine() {
      if (now >= deadline) _;
   }
}
```

Figure 3.20: Solidity Code for the Conditional Start Event Example

3. BPMN Throw Event with the Escalation Event Definition

Another common pattern of halting business process is the use of 'Throw Events'. Throw Events in BPMN will be mapped into revert instruction in Solidity language. Additional information for the error can be provided in the Escalation Code configuration will be used as the revert message.



Event Type	Event Details
Escalation	Escalation 1

Name
Escalation_1_Input

Escalation: Escalation 1

Escalation Code: Error!
Data Type: solidity:string

Figure 3.21: Configuring Message for Escalation Event

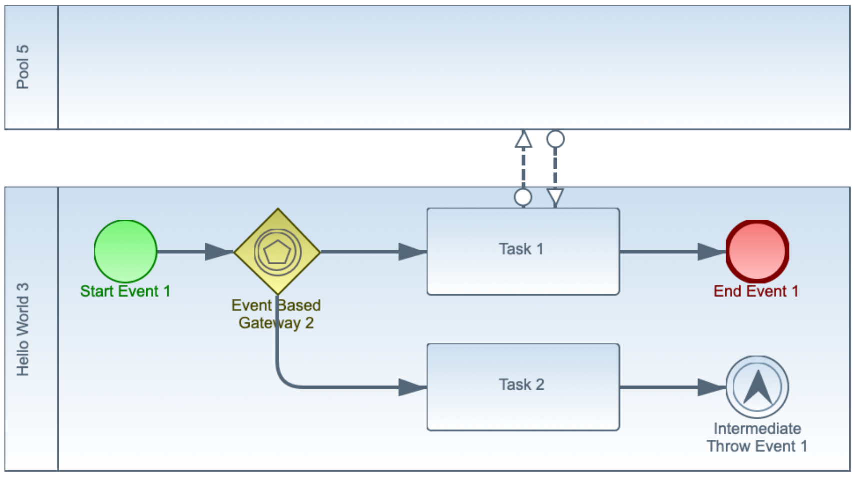


Figure 3.22: Throw Event with Escalation Event Definition Example

Above process model defined in BPMN (Figure 3.22) will generate the below Solidity code depicted in

```
contract HelloWorld3 {
    function task1() public {
    }

    function task2() {
        revert(“Error!”);
    }
```

Figure 3.23;

```
contract HelloWorld3 {
   function task1() public {
   }

   function task2() {
      revert(“Error!”);
   }
```

Generated Solidity Code

```
contract HelloWorld3 {
   function task1() public {
   }

   function task2() {
      revert("Error!");
   }
```

Figure 3.23: Soldity Code for the Throw Event with Escalation Event Definition

(ii) BPMN Tasks

1. **BPMN Tasks (Generic, Manual, User, Business, Service Tasks)**
   All the BPMN tasks (except 'Script', 'Send' and 'Receive' Tasks) shares the same functionality when generating Solidity code. There can be multiple Tasks added in a single request flow chain, the name of the function will be the first name of the first Task of the request flow chain. The first Task that appears after the 'Start Event' is named as Top-Level Task for the internal references.

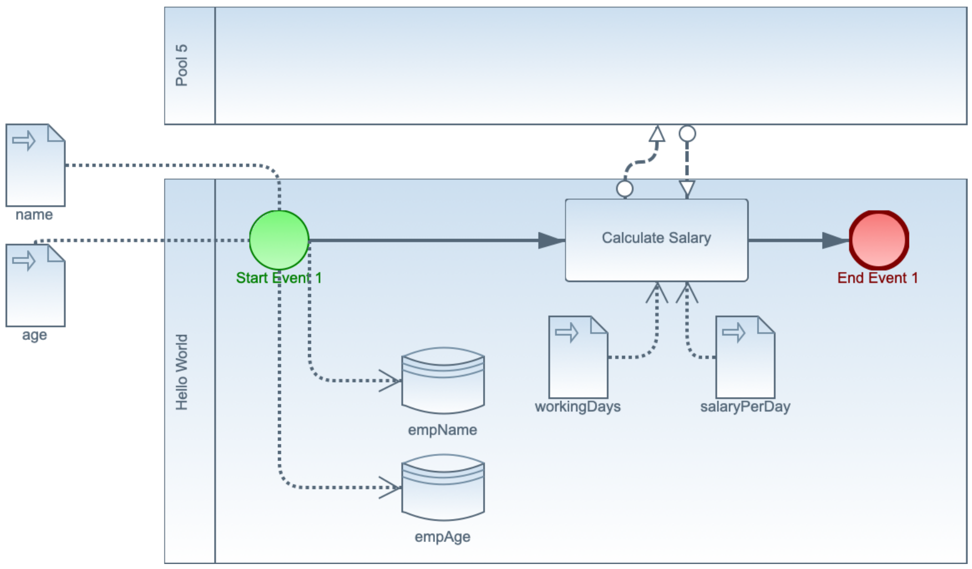


Figure 3.24: Generic Tasks Example

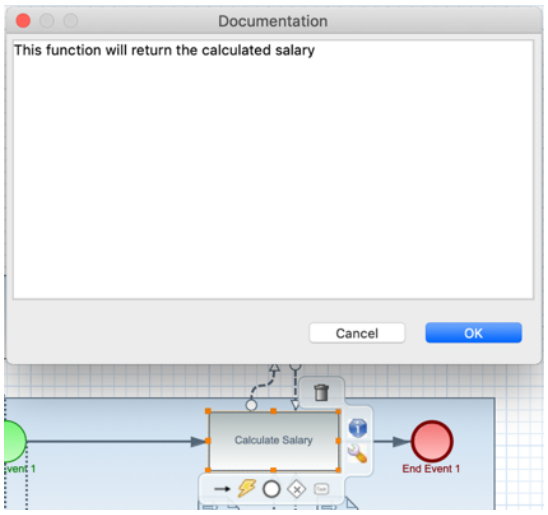


Figure 3.25: Configuring Documentation for BPMN Elements

Generated Solidity Code

```
contract HelloWorld {

    string public empName;
    uint public empAge;

    constructor (string name, uint age) {
        empName = name;
        empAge = age;
    }

    /** This function will return the calculated salary */
    function calculateSalary(uint workingDays, string salaryPerDay)
public returns uint {
        return workingDays * salaryPerDay;
    }
}
```

Figure 3.26: Solidity Code for Tasks Example

2. BPMN Script Task

Code generation for the Script Task is same as the Generic Tasks. Only difference is that the body of the generated function is appended with the 'script value' of the Script Task.

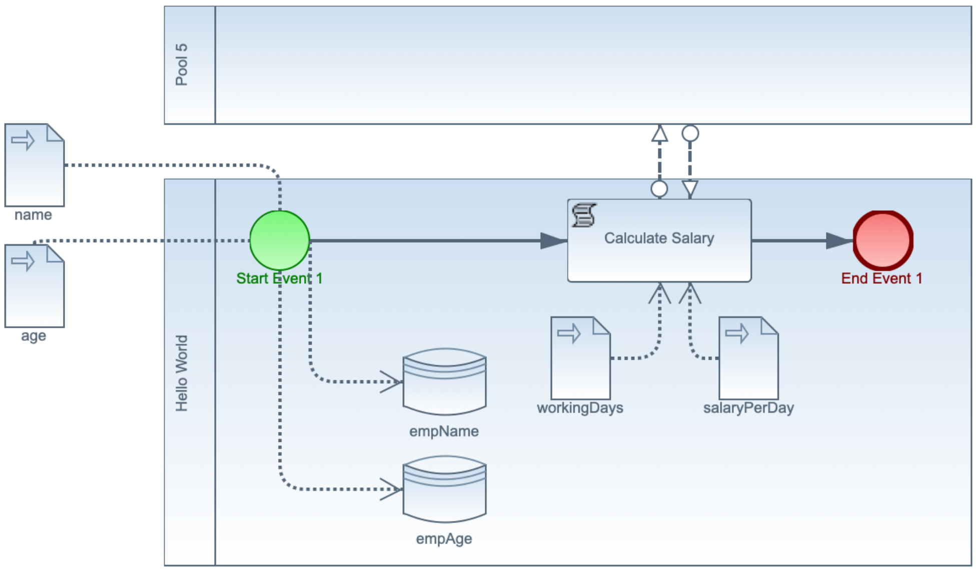
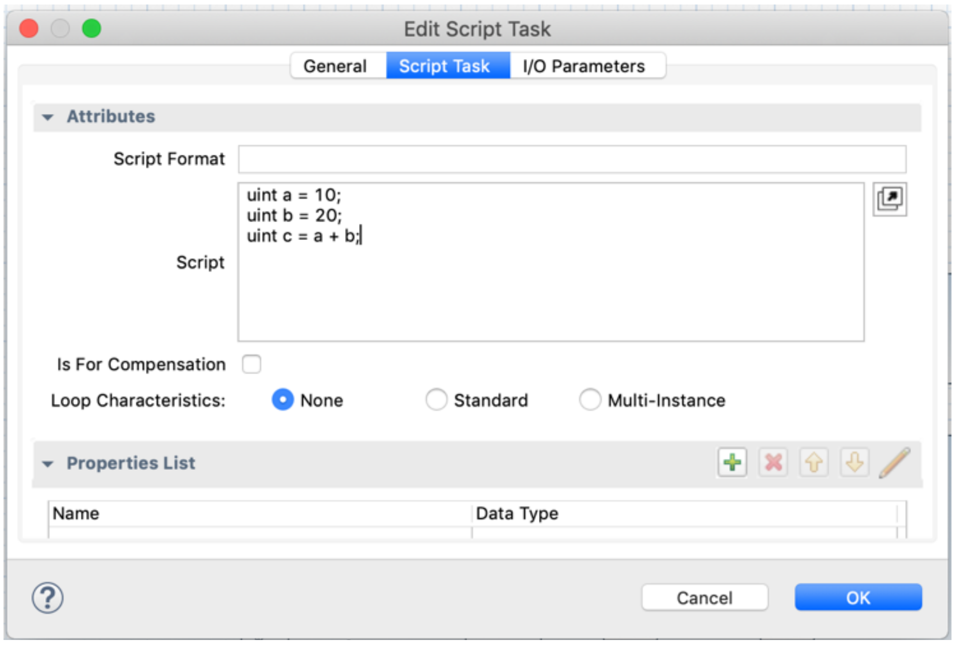


Figure 3.27: Example for Script Task



Script Format:

Script:

```
uint a = 10;
uint b = 20;
uint c = a + b;
```

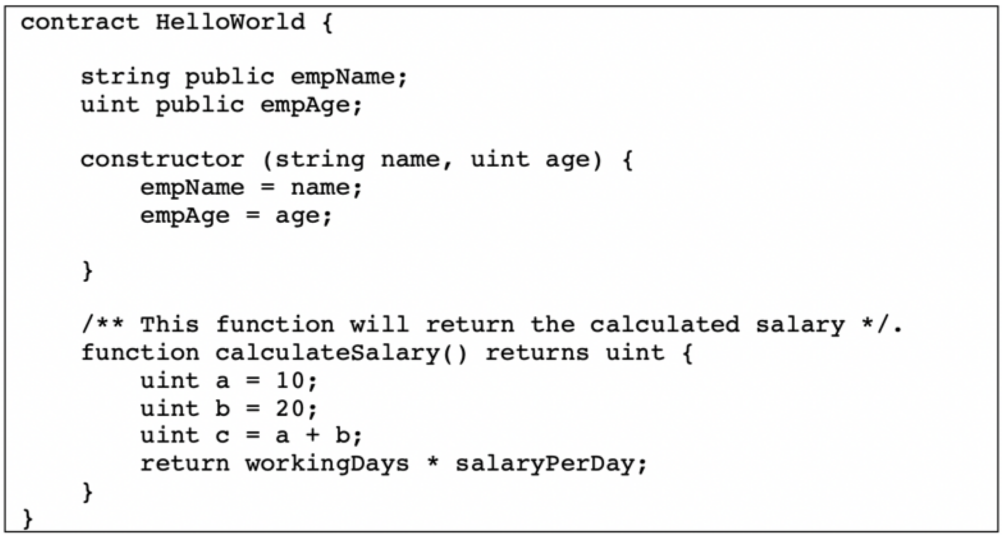
Is For Compensation ☐

Loop Characteristics: [✓ None] Standard Multi-Instance

Name	Data Type

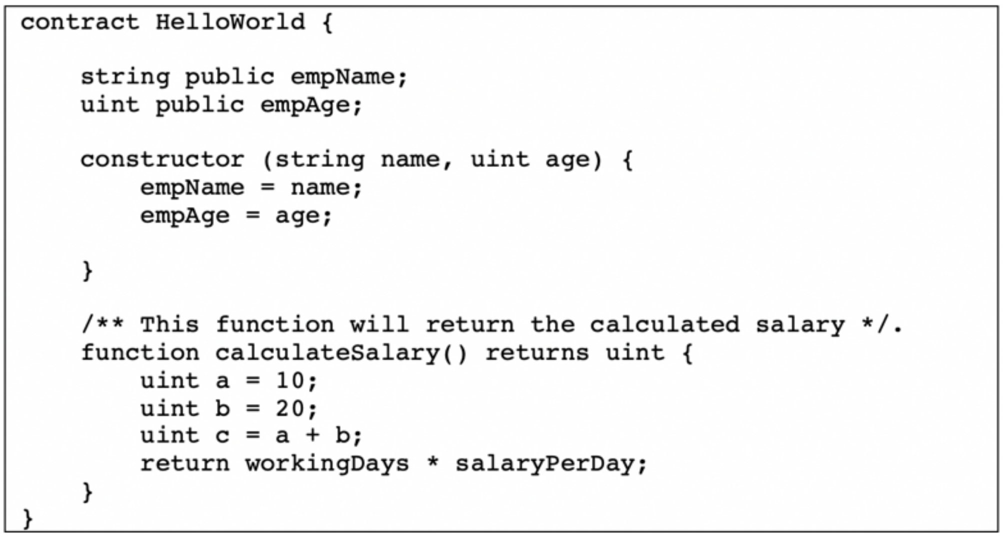
Figure 3.28: Configuring Script Value of the Script Task

Above process model defined in BPMN (Figure 3.27) will generate the below Solidity code depicted in



```
contract HelloWorld {

    string public empName;
    uint public empAge;

    constructor (string name, uint age) {
        empName = name;
        empAge = age;

    }

    /** This function will return the calculated salary */.
    function calculateSalary() returns uint {
        uint a = 10;
        uint b = 20;
        uint c = a + b;
        return workingDays * salaryPerDay;
    }
}
```

Figure 3.29;



```
contract HelloWorld {

    string public empName;
    uint public empAge;

    constructor (string name, uint age) {
        empName = name;
        empAge = age;

    }

    /** This function will return the calculated salary */.
    function calculateSalary() returns uint {
        uint a = 10;
        uint b = 20;
        uint c = a + b;
        return workingDays * salaryPerDay;
    }
}
```

Generated Solidity Code

```
contract HelloWorld {

    string public empName;
    uint public empAge;

    constructor (string name, uint age) {
        empName = name;
        empAge = age;

    }

    /** This function will return the calculated salary */.
    function calculateSalary() returns uint {
        uint a = 10;
        uint b = 20;
        uint c = a + b;
        return workingDays * salaryPerDay;
    }
}
```

Figure 3.29: Solidity Code for Script Task

3. BPMN Send Task

Send Tasks in BPMN is being used for sending messages. When generating Solidity code, Send Tasks are mapped into notifications in Solidity. All the input parameters for the BPMN Message element will be used for the event definition and parameter values will be used for the 'emit' instruction in Solidity.

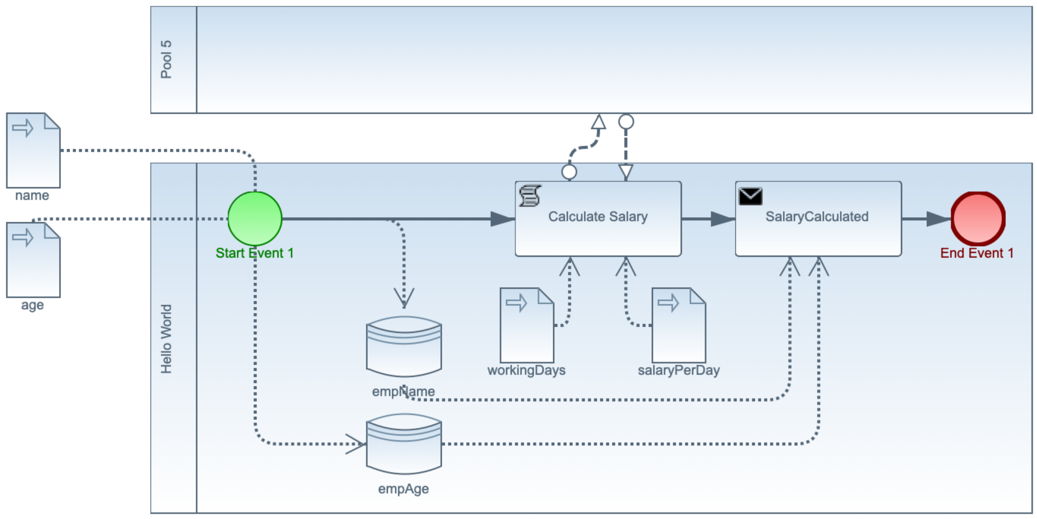
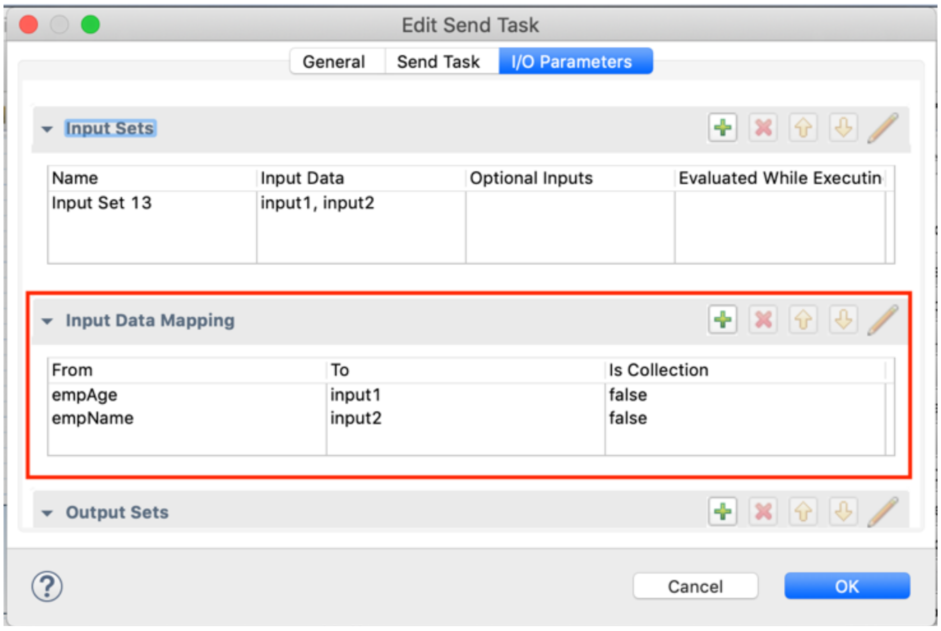


Figure 3.30: Example for Send Task



Name	Input Data	Optional Inputs	Evaluated While Executin
Input Set 13	input1, input2		

From	To	Is Collection
empAge	input1	false
empName	input2	false

Figure 3.31: Configuring Input Data Mapping for Send Task

Above process model defined in BPMN (Figure 3.30) will generate the below Solidity code depicted in Figure 3.32;

<u>**Generated Solidity Code**</u>

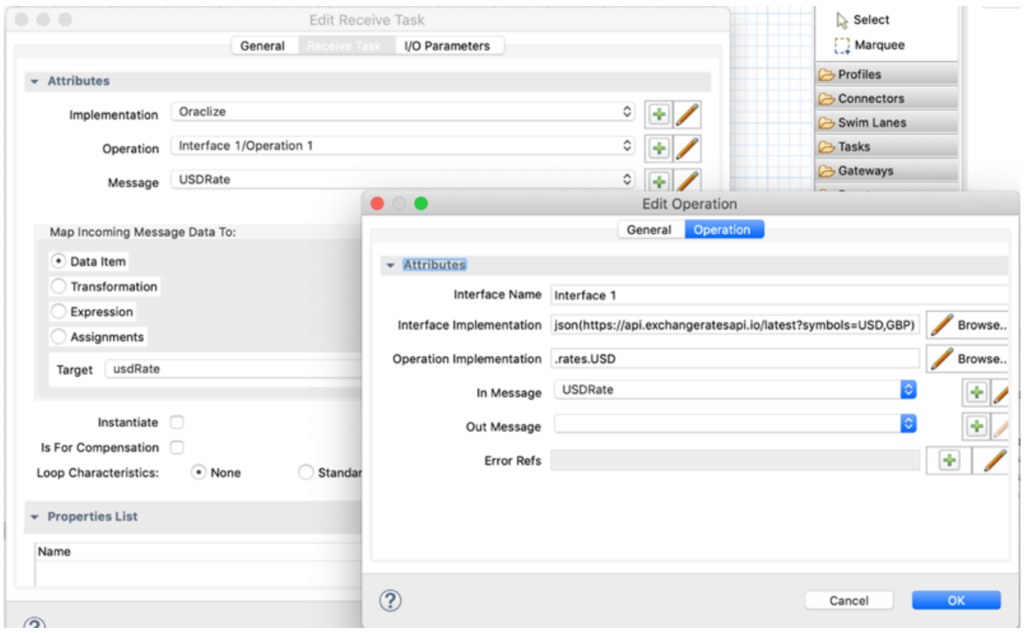
```
contract HelloWorld {

    string public empName;
    string public empAge;

    event SalaryCalculated(string empName, uint empAge);

    constructor (string name, uint age) {
        empName = name;
        empAge = age;
    }

    /** This function will return the calculated salary */
    function calculateSalary() public returns uint {
        uint a = 10;
        uint b = 20;
        uint c = a + b;
        emit SalaryCalculated(empName, empAge);
        return workingDays * salaryPerDay;
    }
}
```

Figure 3.32: Solidity Code for Send Task Example

4. BPMN Receive Task

Receive Task in BPMN is used to receive incoming messages from the external parties. When generating Solidity code for the Receive

Tasks are mapped into an API call for an oracle node off the chain. Potential uses of these API calls are to retrieve the latest currency exchange rates, latest fuel prices…etc. In the implementation level, Oraclize library is used to perform these operations.


Edit Receive Task

General | Receive Task | I/O Parameters

Attributes

Implementation: Oraclize
Operation: Interface 1/Operation 1
Message: USDRate

Map Incoming Message Data To:
- [✓ Data Item]
- Transformation
- Expression
- Assignments

Target: usdRate

Instantiate: ☐
Is For Compensation: ☐
Loop Characteristics: [✓ None] Standar

Properties List

Name

Edit Operation

General | Operation

Attributes

Interface Name: Interface 1
Interface Implementation: json(https://api.exchangeratesapi.io/latest?symbols=USD,GBP)
Operation Implementation: .rates.USD
In Message: USDRate
Out Message:
Error Refs:

Figure 3.33: Configuring Incoming Message for Receive Task

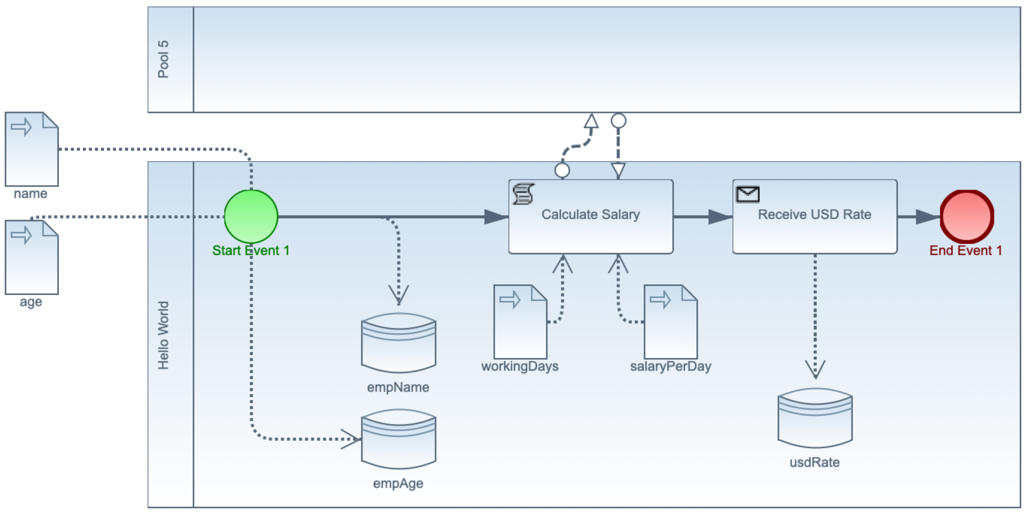


Figure 3.34: Example for Receive Task

Above process model defined in BPMN (Figure 3.33) will generate the below Solidity code depicted in Figure 3.35;

<u>Generated Solidity Code</u>

```
import "github.com/oraclize/ethereum-api/oraclizeAPI.sol";

contract HelloWorld is usingOraclize {

    string public empName;
    string public empAge;
    string public usdRate;

    constructor (string name, uint age) {
        empName = name;
        empAge = age;
    }

    /** This function will return the calculated salary */
    function calculateSalary() public returns uint {
        uint a = 10;
        uint b = 20;
        uint c = a + b;
        oraclize_query("URL",
'json(https://api.exchangeratesapi.io/latest?symbols=USD,GBP).rates.USD');
        return workingDays * salaryPerDay;
    }

    function __callback(bytes32 _myid, string memory _result) public {
        require(msg.sender == oraclize_cbAddress());
        usdRate = _result;
    }
}
```
Figure 3.35: Solidity Code for the Receive Task

(iii) BPMN Data Objects

Another set of important BPMN elements are 'Data Objects' and 'Data Store References'. When mapping into the Solidity code, 'Data Objects' are mapped into private state variables and 'Data Store References' are mapped into public state variables. If a 'Data Object' or 'Data Store Reference' has one or more 'Data Inputs' it will be mapped as a struct in Solidity language.

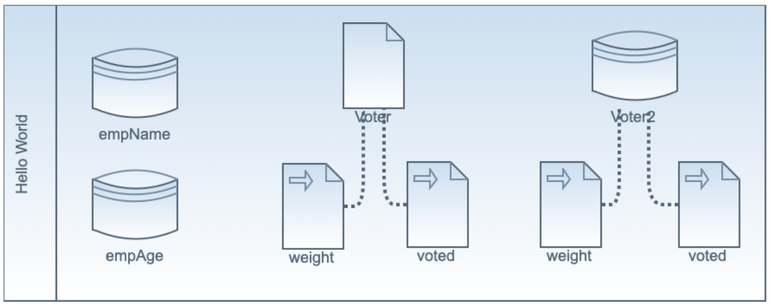


Figure 3.36: Data Objects and Data Store References Example

Above process model defined in BPMN (Figure 3.36) will generate the below Solidity code depicted in Figure 3.37.

Generated Solidity Code

```
contract HelloWorld {
    struct Voter2 public {
        uint weight;
        bool voted;
    }
    struct Voter {
        bool voted;
        uint weight;
    }
    string public empName;
    string public empAge;
}
```

Figure 3.37: Solidity Code for Data Objects and Data Store References Example

3.4.3. BPMN Representation of Solidity Top Level Constructs

In this analysis, Solidity code express-ability using BPMN notations is evaluated. An example of using top-level constructs in Solidity is depicted in Figure 3.38. The genereted Solidity code for the business process model is in Figure 3.39.

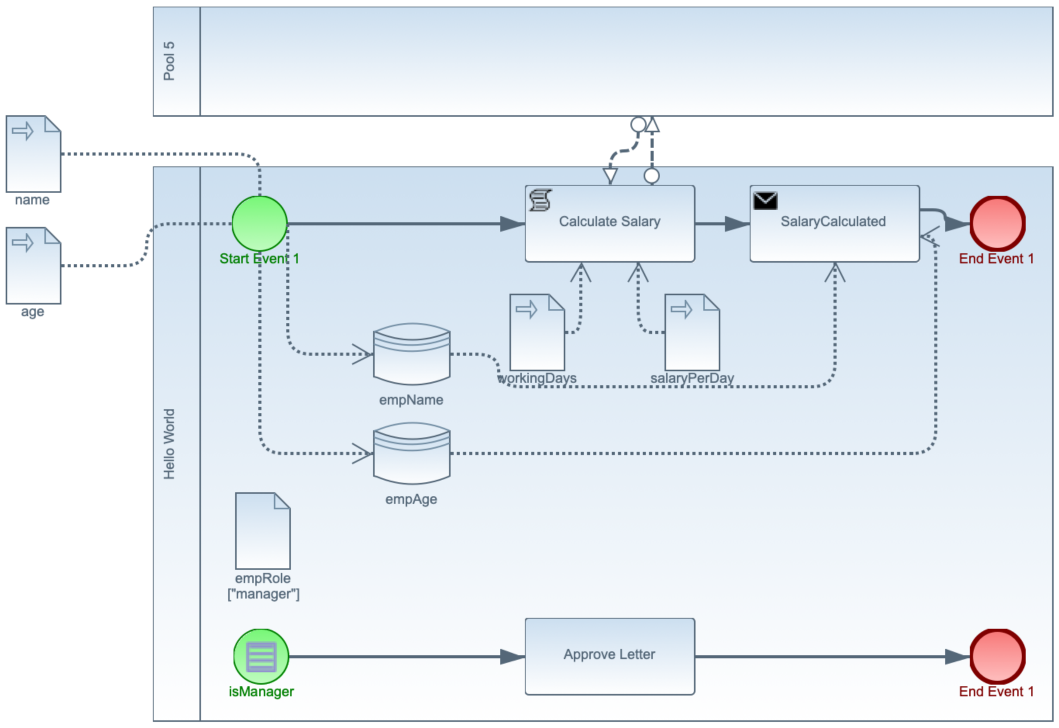


Figure 3.38: Solidity Top-Level Constructs Example

```
contract HelloWorld {
    string public empName;
    string public empAge;
    string empRole = “manager”;
    event SalaryCalculated(string empName, uint empAge);
    constructor (string name, uint age) {
        empName = name;
        empAge = age;
    }
    function approveLetter() isManager {
    }
    modifier isManager() {
        if(empRole == “manager”) _
    }
    /** This function will return the calculated salary */
    function calculateSalary() public returns uint {
        uint a = 10;
        uint b = 20;
        uint c = a + b;
        emit SalaryCalculated(empName, empAge);
        return workingDays * salaryPerDay;
    }
}
```

Figure 3.39: Solidity Code for the Top-Level Constructs Example

(i) Solidity Contract

In order to represent a Contract in BPMN, A single **Participant** (or **Pool**) can be utilized. Pre-processed name of the participant will be mapped as the name of the contract. For the pre-processing, all whitespaces are removed and words are joined with camel-case.

However, the name of the participant should not be prefixed with the 'Library' or 'Interface' as they are already reserved words.

(ii) Solidity Interface / Library

Interface or library in Solidity will be represented as a **Participant** in BPMN with name prefixed by 'Library' or 'Interface' accordingly.

(iii) Solidity State Variables

State variables can be public or private. If it is a public state variable, **Data Stores** in BPMN can be used to represent it. By nature, Data Stores is accessible by any process. On the other hand, a non-public state variable can be represented using **Data Objects**. By nature, Data Objects has the life span of a particular process that it resides.

(iv) Solidity Functions

Solidity functions can be represented as Tasks. In generic, top-level Tasks (which are first task in a single request flow chain) are mapped into functions. Pre-processed name of the top-level task will be mapped into **function-name**. Any documentation provided in top-level task will be added a **code-documentation** for the particular **function**. Data Inputs with data associations will be mapped as parameters of the **function-signature**. Statements for the **function-body** is generated by traversing through the sequence flows outgoing through the top-level task, and whenever there's an output mapping with the reserved keyword 'return' it will re-ordered into bottom of the function body. Whenever, there's an explicit 'return', Data Type of the output mapping is used as the **return-type** of the function-signature. If nothing specified, return type will be 'void' by default. Further, adding ':<modifier>' in function name will be added as a **modifier** in the function-signature. For instance, a Task with the name 'Task1| : pure' will result a function with the name 'task1( )' and also an inline modifier 'pure' will be added into the function-signature.

(v) Solidity Function Modifiers

Any number of pre-defined function modifiers can be added into the function-signature by adding ':<modifier>' after the function-name. A custom function modifier is also possible with the Conditional Start Events. In this case, an implementation of the custom modifier will be added into top-level of the contract and modifier will be added into any immediate Tasks after the event.

(vi) **Solidity Events**
Event notifications in Solidity can be represented as a Send Task in BPMN. Input data mapping configurations can be used to define the event structure of the even-definition and event parameters for the 'emit' Solidity instruction as well.

(vii) **Solidity Structs**
Structs can be represented as Data Stores and Data Objects with one or more Data Inputs associated as fields of the struct. Public Structs can be represented as Data Stores and non-public structs can be represented in Data Objects.

(viii) **Solidity Enums**
Enums can be represented as a Data Type of the Data Object and Data Stores. Enum value constants can be defined in the Data State of the Data Object of Data Store.

(ix) **Solidity Code Documentation**
Code documentation for a particular Solidity struct is represented using the 'documentation' attribute of the respective BPMN element.

4. EVALUATION AND RESULTS

4.1. Translator Evaluation

A translator which converts one source language in to another source language is called a Transpiler. Evaluating transpilers is challenging and tedious task. The evaluation of the translator is performed in two approaches (1) Matching a set of abstract features in BPMN and Solidity, (2) Conducting a survey for a selected user group of expertise.

4.1.1. Abstract Features Matching in BPMN and Solidity

Related work in the same domain [19] [17] has used similar approaches for the evaluation of the translation. A collection of abstract features is compared against the input and output source languages.

In the data collection phase of this study, a sequence of smart contracts written in Solidity language was gathered. These are mapped into business process models using selected BPMN process modeler, Eclipse BPMN 2.0 plugin. Resulting process models could be transformed in back to the Solidity smart contracts with a minimum effort and this enables minimizing the effect of transferability factor from the process models.

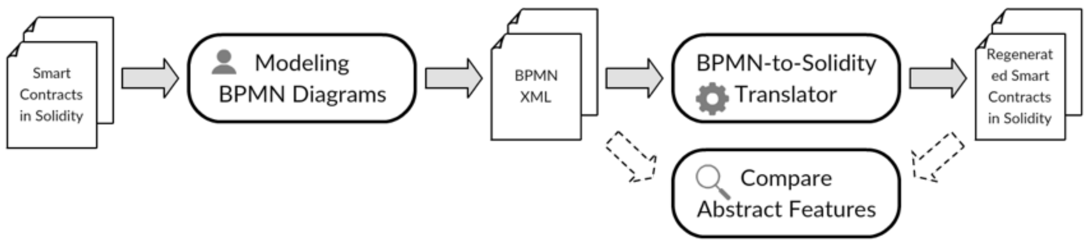

Smart Contracts in Solidity
Modeling BPMN Diagrams
BPMN XML
BPMN-to-Solidity Translator
Regenerated Smart Contracts in Solidity
Compare Abstract Features

Figure 4.1: BPMN to Solidity Translator Evaluation Using Abstract Features

Thereafter, BPMN process models created from the early process is fed into the translator to produce Solidity smart contracts. Subsequent, generated Solidity smart contracts and process models defined in BPMN XML are matched against a set of abstract features as depicted in Figure 4.1. A list of abstract features and meaning of each feature in BPMN and Solidity language is listed in Table 4.1.

Table 4.1: A List of Abstract Features in BPMN and Solidity

Abstract Features	BPMN XML	Solidity Code
1. **Entry Points**	The top-level BPMN Tasks of the BPMN Processes	Function definitions
2. **Exclusive Branch Points**	The locations in the BPMN process where alternatives are based on the conditions within the outgoing sequence flows and only one alternative is chosen	The 'If' conditions reside within the body of the Solidity Function
3. **Persistent Points**	BPMN Data Stores	The State Variables

As per the literature review, a similar evaluation approaches are being used for the evaluation of the similar systems [19] [17]. For the purpose of evaluation, Weber, Ingo et. al exploits a sequence of permissible execution traces for each process model. The expected outcome of the BPMN-to-Java transformer in [19] uses a sequence of defective instances as for example zero errors, incomplete diagrams, element misses and isolated elements for the evaluation.

Results of the abstract feature matching in BPMN and Solidity is summarized in Table 4.2. Six (6) different process models and respective regenerated Solidity code in different domains and scales are tested against sixty distinct process model permutations (6 x 10 alterations). Impacts and existence of the abstract features in each case is manually verified.

Table 4.2: Summary of Abstract Feature Matching in BPMN XML and Solidity

Contract Name	BPMN Diagrams (same values for the re-generated Solidity codes)			Transform Time (ms)
	Entry Points	**Branch Points**	**Persist. Points**	
Rental Agreement	3	0	9	272ms
Wedding Gift	2	0	3	221ms
Sellable	4	0	4	283ms
Lottery	5	1	3	257ms
Basic Token	2	0	2	247ms
Crowd Sale	3	6	5	304ms

4.1.2. Conducting a Survey for a Selected User Group

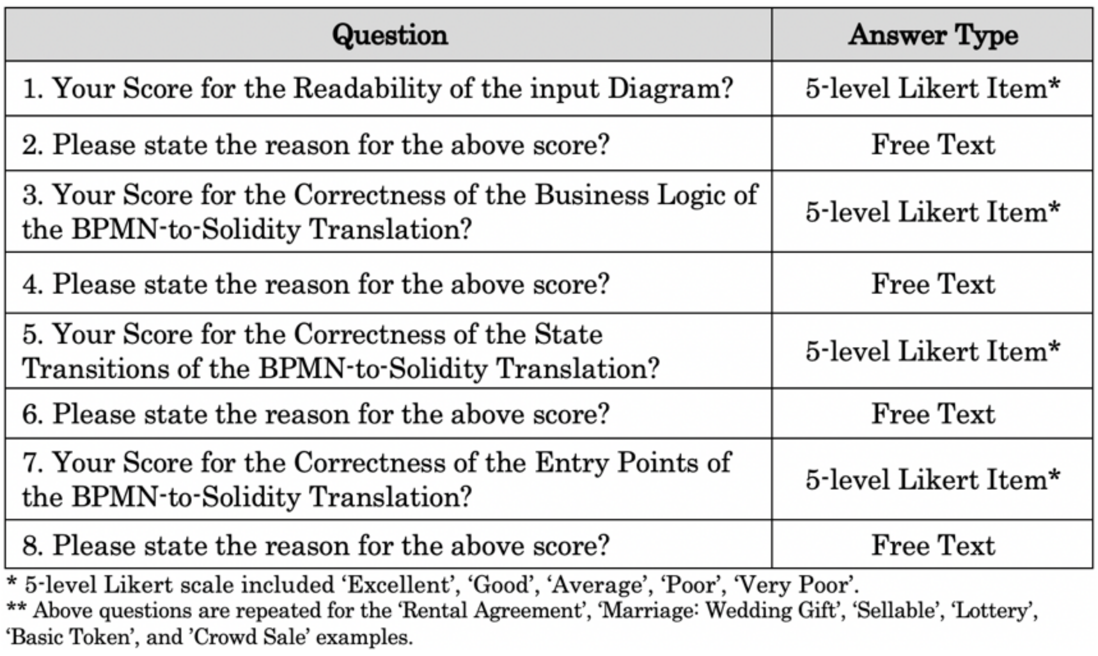
A survey was conducted to measure the level of accuracy of the BPMN-to-Solidity translator for a selected user group of expertise. The survey is prepared with an online questionnaire[8] with two main sections; 1) Details of the Participant (refer Table 4.3) and 2) Examples Input and Output of the Translator (refer Table 4.4). The reason for selecting Likert-Scale items for the correctness of the translation is that there's no single 'Yes/No' answer for them since it is a language translation from a source language (BPMN XML) to another target language (Solidity language).

Table 4.3: Details of the Participant

Question	Answer Type
1. Your Name?	Free Text
2. Your Workplace / Academic Institution?	Free Text
3. Your Current Position / Profession?	Free Text
4. Rate Your knowledge on BPMN?	5-level Likert Item*
5. Rate Your knowledge on Programming Skills?	5-level Likert Item*
6. Rate Your knowledge on Solidity Language?	5-level Likert Item*

* 5-level Likert scale included 'Excellent', 'Good', 'Average', 'Poor', 'Very Poor'.

Table 4.4: Example Input and Output of the Translator



Question	Answer Type
1. Your Score for the Readability of the input Diagram?	5-level Likert Item*
2. Please state the reason for the above score?	Free Text
3. Your Score for the Correctness of the Business Logic of the BPMN-to-Solidity Translation?	5-level Likert Item*
4. Please state the reason for the above score?	Free Text
5. Your Score for the Correctness of the State Transitions of the BPMN-to-Solidity Translation?	5-level Likert Item*
6. Please state the reason for the above score?	Free Text
7. Your Score for the Correctness of the Entry Points of the BPMN-to-Solidity Translation?	5-level Likert Item*
8. Please state the reason for the above score?	Free Text

* 5-level Likert scale included 'Excellent', 'Good', 'Average', 'Poor', 'Very Poor'.
** Above questions are repeated for the 'Rental Agreement', 'Marriage: Wedding Gift', 'Sellable', 'Lottery', 'Basic Token', and 'Crowd Sale' examples.

[8] The online questionnaire used to collect responses is available in https://forms.gle/jG4yAc8uBRq2ZgVQ8

The participants of the Survey include different categories such as Business Analysts, Technical Specialists, Software Engineer, Masters Graduates, Lecturer, and Ph.D. Students. Even though the sample size is quite low(N=7), it serves the goal of the study as to get the feedback of an expertise user group. Thus, the reliability of the feedback is believed to be increased.

1. Rate your knowledge on BPMN?

The summarized results for this question are depicted in Figure 4.2. More than 86% percent has BPMN knowledge and only 14% has 'Poor' knowledge in BPMN.

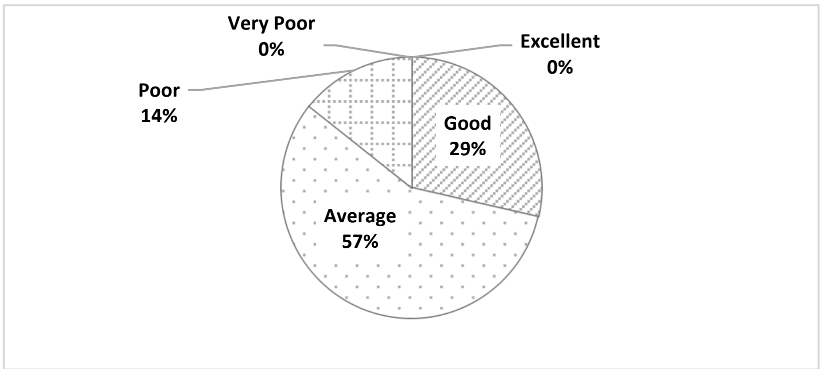


Figure 4.2: Summary of the BPMN Knowledge Ratings

2. Rate your knowledge on Programming Skills?

The summarized results for this question are depicted in Figure 4.3. As per the results all participants have 'Average', 'Good' and 'Excellent' programming skills. No ratings for 'Poor' or the 'Very Poor' level of knowledge.

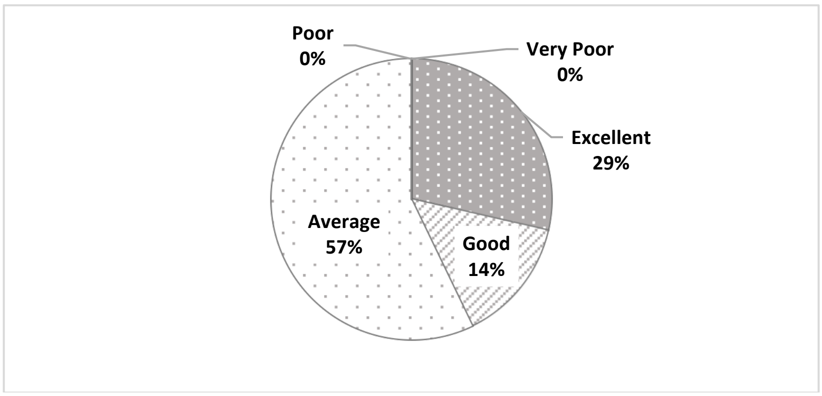


Figure 4.3: Summary of the Programming Skills Ratings

3. Rate your knowledge on Solidity Language?

The summarized results for this question are depicted in Figure 4.4. As per the results 29% of the participants are 'Very Poor' in Solidity language. It is also important to highlight that there are no participants with the 'Excellent' and 'Good' in knowledge. This is expectable since smart-contract development is a new area for the most of the Developers.

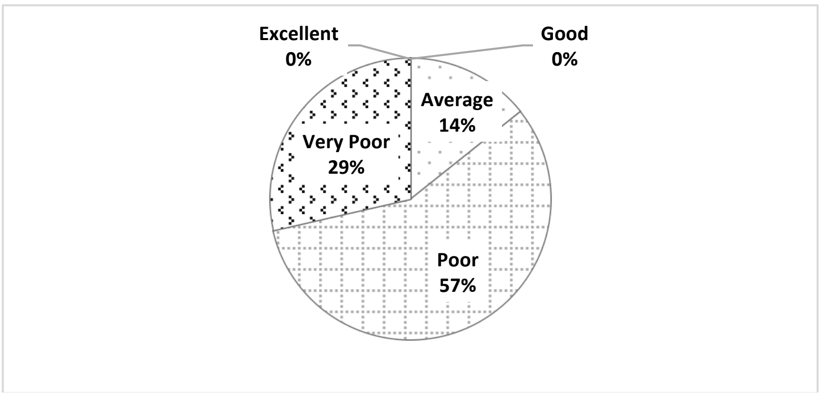


Figure 4.4: Summary of the Solidity Language Ratings

Following are the overall summary of the results for the four questions provided.

4. Your score for the Readability of the input Diagram?

In Figure 4.5, 88% (24+35+29) of the participants agrees that the input BPMN diagram notations are sufficient and only 12% claims it is 'Poor' and need to be improved. However, there are no participants with 'Very Poor' status.

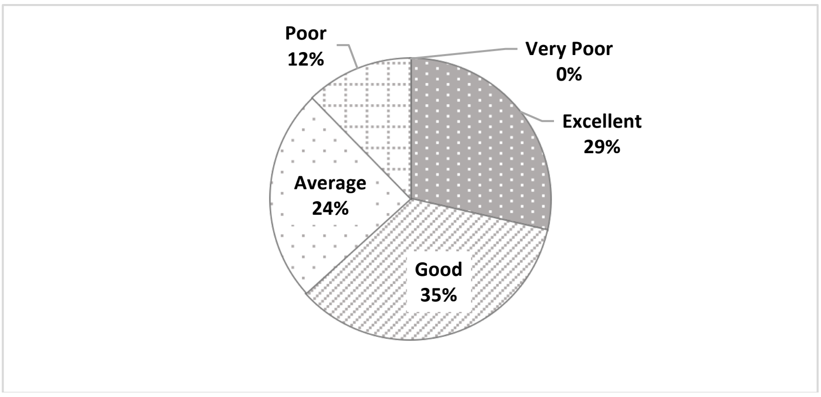


Figure 4.5: Summary of the Readability of the Input Diagram

Some of the free-text feedback received for this question includes below;

- “The language used for defining objects is understandable”
- “Quiet clear and relationships are well illustrated”
- “Isolated notations, unnamed relationships”
- “Its similar to a flow diagram?”
- “what is pool 1?”

5. Your score for the Correctness of the Business Logic of the BPMN-to-Solidity Translation?

In Figure 4.6, 24% of the participants believes that the business logic of the translation is ‘Excellent’ and 90% (40+26+24) of the participants agrees that the business logics of the translation is sufficiently accurate. Only 10% mentions the translation is ‘Poor’ and need improvements. However, there’s none for the ‘Very Poor’.

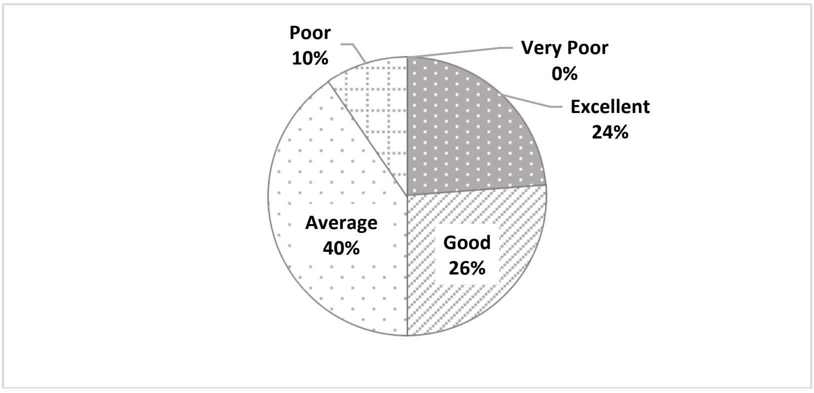


Figure 4.6: Summary of the Correctness of the Business Logic

Some of the free-text feedback received for this question includes below;

- “State transitions are correctly stated”
- “Number of logics aligns with the diagram”
- “couldn’t figure out the construct”
- “I think the mapping almost covered the given scenarios”
- “Good conversion. Small gaps in readability”
- “Decision point can be improved”
- “The hidden details of BPMN is clearly expressed in the code.”

6. Your Score for the Correctness of the State Transitions of the BPMN-to-Solidity Translation?

In Figure 4.7, none of participants rated 'Very Poor' for the correctness of the State Transitions. 19% stated as 'Excellent' and 74% (26+29+19) of the participants are satisfied by the correctness of the translation.

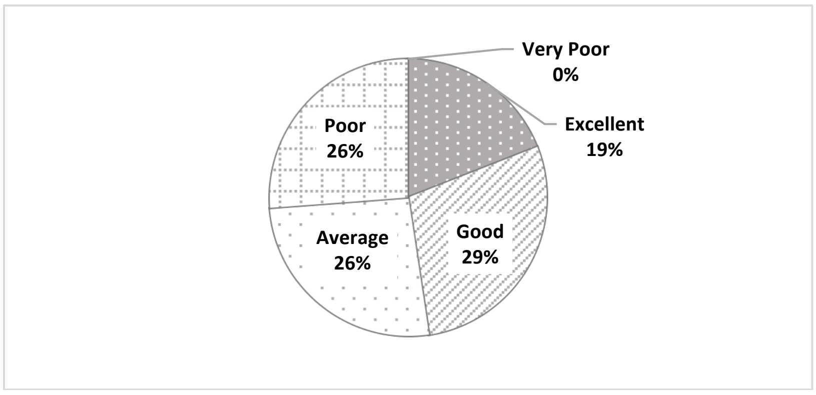


Figure 4.7: Summary of the Correctness of the State Transitions

Some of the free-text feedback received for this question includes below;

- "Good translation - e.g. variable names"
- "State transitions have been correctly modeled"
- "I can identify the mapping of variables and methods between BPMN-to-Solidity to a greater extent"
- "Most of the state transitions are captured, yet can improve"
- "No clear enough representations"
- "state transition is not playing major part in the diagram or in the code in this scenario"
- "program doesn't depend much on state transition"

7. Your Score for the Correctness of the Entry Points of the BPMN-to-Solidity Translation?

In Figure 4.8, none of participants rated, 'Very Poor'. 14% stated as 'Excellent' and 83% (29+40+14) of participants are satisfied with the correctness of the entry points of the translation.

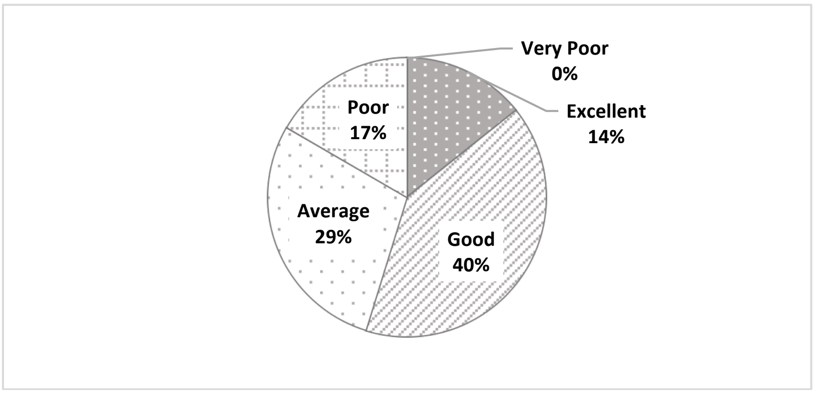


Figure 4.8: Summary of the Correctness of the Entry Points

Some of the free-text feedback received for this question includes below;

- “public methods are properly defined in BPMN as well as in Solidity.”
- “Clearly marked in both BPMN and Solidity”
- “Got the logic”
- “All entry points are represented in the translation”
- “not much familiar”
- “Can be identified to some extent, but I feel the diagram should be more detailed”
- “Function wise decoupling has achieved this.”
- “Could improve the representations of the entry points”

Further, the detailed responses for the survey can be found in Appendix E in page 150. These results suggest that the translator was able to translate the BPMN diagram XML to Solidity smart contracts in a satisfactory level.

4.2. Re-generated Solidity Smart Contract Code Evaluation

Truffle[9] is a framework that provides a development environment for building, testing and deploying applications on blockchains using Etherium Virtual Machine (EVM). Re-generated Solidity smart contracts were tested using tests written in Truffle framework. Tests are written for a selected set of smart contracts namely, 'Lottery', 'Marriage-Wedding Gifts', 'Basic Token' which covers most frequent set of BPMN elements [20]. Input process models defined in BPMN and generated Solidity code is available in section **Error! Reference source not found.** (page **Error! Bookmark not defined.**). Additional configurations performed for the test framework is listed below;

migrations/deploy_contracts.js

```
const BasicToken = artifacts.require("BasicToken");
const Marriage = artifacts.require("Marriage");
const Lottery10Users = artifacts.require("Lottery10Users");


module.exports = function(deployer, networks, accounts) {
  deployer.deploy(Lottery10Users);
  deployer.deploy(BasicToken, 1000);
  deployer.deploy(Marriage, accounts[2], accounts[3]);
};
```

truffle-config.js

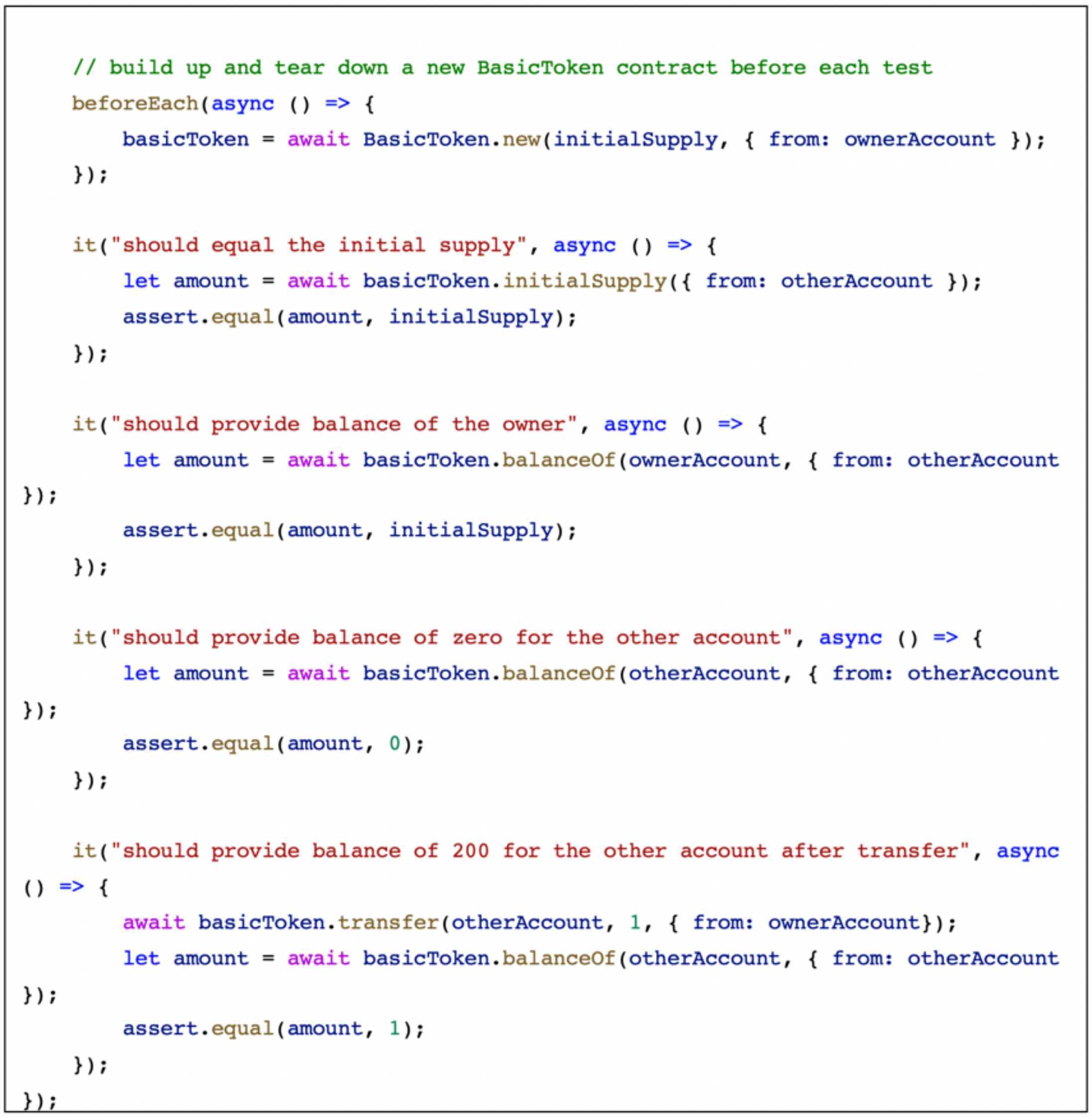
```
mocha: {
    // timeout: 100000
    reporter: 'eth-gas-reporter'
  },
```

A sample test written in truffle is listed below. Truffle tests for the other selected smart contracts is available Appendix D (page 147).

tests/BasicToken.test.js

[9] Truffle development environment and test framework is available in https://www.trufflesuite.com/docs/truffle/testing/testing-your-contracts online, accessed 19-01-2020.

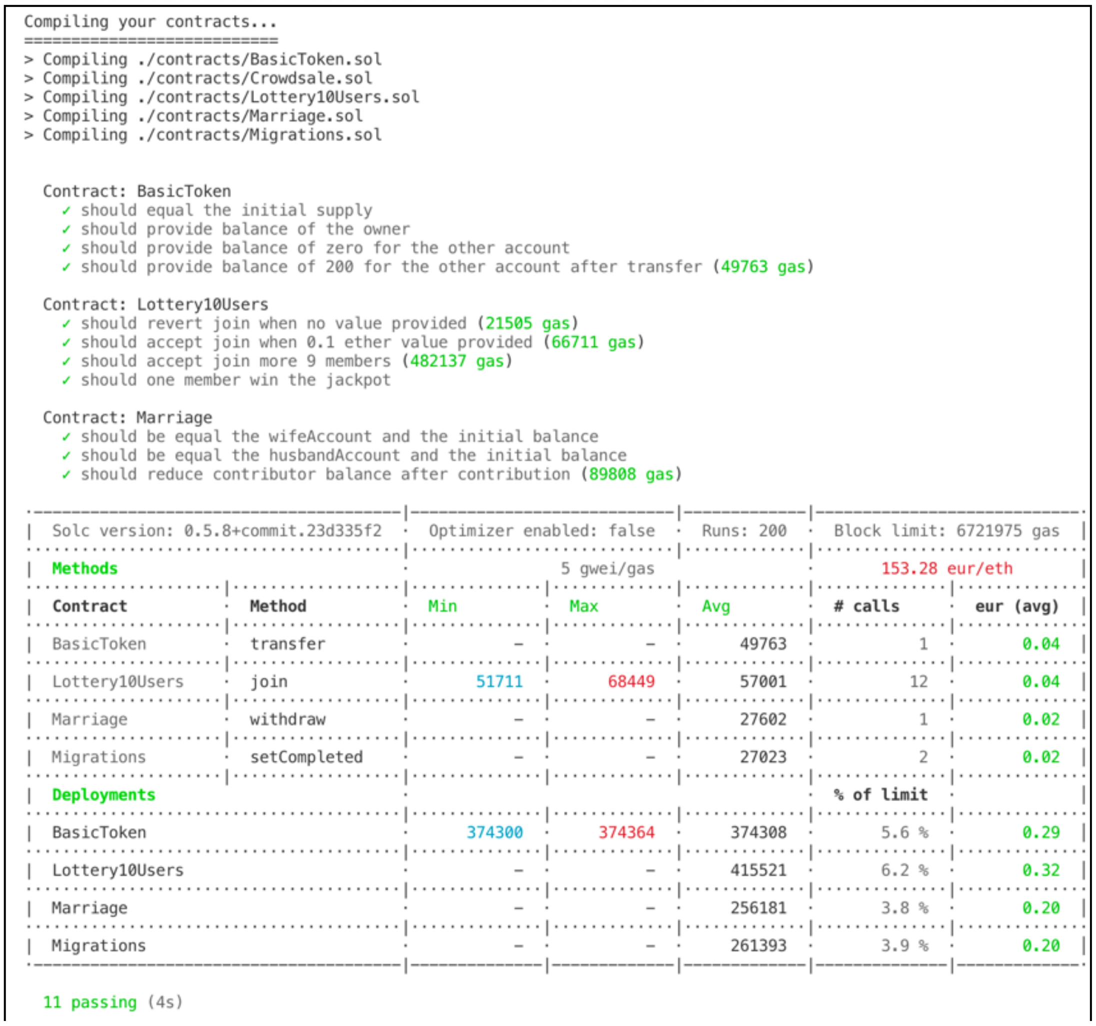
```
    // build up and tear down a new BasicToken contract before each test
    beforeEach(async () => {
        basicToken = await BasicToken.new(initialSupply, { from: ownerAccount });
    });

    it("should equal the initial supply", async () => {
        let amount = await basicToken.initialSupply({ from: otherAccount });
        assert.equal(amount, initialSupply);
    });

    it("should provide balance of the owner", async () => {
        let amount = await basicToken.balanceOf(ownerAccount, { from: otherAccount
});
        assert.equal(amount, initialSupply);
    });

    it("should provide balance of zero for the other account", async () => {
        let amount = await basicToken.balanceOf(otherAccount, { from: otherAccount
});
        assert.equal(amount, 0);
    });

    it("should provide balance of 200 for the other account after transfer", async
() => {
        await basicToken.transfer(otherAccount, 1, { from: ownerAccount});
        let amount = await basicToken.balanceOf(otherAccount, { from: otherAccount
});
        assert.equal(amount, 1);
    });
});
```



```
    // build up and tear down a new BasicToken contract before each test
    beforeEach(async () => {
        basicToken = await BasicToken.new(initialSupply, { from: ownerAccount });
    });

    it("should equal the initial supply", async () => {
        let amount = await basicToken.initialSupply({ from: otherAccount });
        assert.equal(amount, initialSupply);
    });

    it("should provide balance of the owner", async () => {
        let amount = await basicToken.balanceOf(ownerAccount, { from: otherAccount
});
        assert.equal(amount, initialSupply);
    });

    it("should provide balance of zero for the other account", async () => {
        let amount = await basicToken.balanceOf(otherAccount, { from: otherAccount
});
        assert.equal(amount, 0);
    });

    it("should provide balance of 200 for the other account after transfer", async
() => {
        await basicToken.transfer(otherAccount, 1, { from: ownerAccount});
        let amount = await basicToken.balanceOf(otherAccount, { from: otherAccount
});
        assert.equal(amount, 1);
    });
});
```

Figure 4.9: Truffle Test Written for the Basic Token Smart Contract

Execution output of truffle test cases for the selected smart contracts is depicted in Figure 4.10.



```
Compiling your contracts...
===========================
> Compiling ./contracts/BasicToken.sol
> Compiling ./contracts/Crowdsale.sol
> Compiling ./contracts/Lottery10Users.sol
> Compiling ./contracts/Marriage.sol
> Compiling ./contracts/Migrations.sol


  Contract: BasicToken
    ✓ should equal the initial supply
    ✓ should provide balance of the owner
    ✓ should provide balance of zero for the other account
    ✓ should provide balance of 200 for the other account after transfer (49763 gas)

  Contract: Lottery10Users
    ✓ should revert join when no value provided (21505 gas)
    ✓ should accept join when 0.1 ether value provided (66711 gas)
    ✓ should accept join more 9 members (482137 gas)
    ✓ should one member win the jackpot

  Contract: Marriage
    ✓ should be equal the wifeAccount and the initial balance
    ✓ should be equal the husbandAccount and the initial balance
    ✓ should reduce contributor balance after contribution (89808 gas)

·--------------------------------------|--------------------------|------------|--------------------------·
|  Solc version: 0.5.8+commit.23d335f2  ·  Optimizer enabled: false  ·  Runs: 200  ·  Block limit: 6721975 gas  |
·······································|··························|············|···························
|  Methods                             ·                5 gwei/gas              ·       153.28 eur/eth       |
····················|··················|·············|············|············|·············|·············
|  Contract         ·  Method          ·  Min        ·  Max       ·  Avg       ·  # calls    ·  eur (avg)  |
····················|··················|·············|············|············|·············|·············
|  BasicToken       ·  transfer        ·          -  ·         -  ·     49763  ·          1  ·       0.04  |
····················|··················|·············|············|············|·············|·············
|  Lottery10Users   ·  join            ·      51711  ·     68449  ·     57001  ·         12  ·       0.04  |
····················|··················|·············|············|············|·············|·············
|  Marriage         ·  withdraw        ·          -  ·         -  ·     27602  ·          1  ·       0.02  |
····················|··················|·············|············|············|·············|·············
|  Migrations       ·  setCompleted    ·          -  ·         -  ·     27023  ·          2  ·       0.02  |
····················|··················|·············|············|············|·············|·············
|  Deployments                         ·                                       ·  % of limit ·             |
·······································|·············|············|············|·············|·············
|  BasicToken                          ·     374300  ·    374364  ·    374308  ·      5.6 %  ·       0.29  |
·······································|·············|············|············|·············|·············
|  Lottery10Users                      ·          -  ·         -  ·    415521  ·      6.2 %  ·       0.32  |
·······································|·············|············|············|·············|·············
|  Marriage                            ·          -  ·         -  ·    256181  ·      3.8 %  ·       0.20  |
·······································|·············|············|············|·············|·············
|  Migrations                          ·          -  ·         -  ·    261393  ·      3.9 %  ·       0.20  |
·--------------------------------------|-------------|------------|------------|-------------|-------------·

  11 passing (4s)
```

Figure 4.9: Test Execution Output of the Tests Written in Truffle

BPMN elements covered in each process model of the contracts is listed in **Error! Reference source not found.**. Basic control patterns and most frequent BPMN elements are covered as in [20].

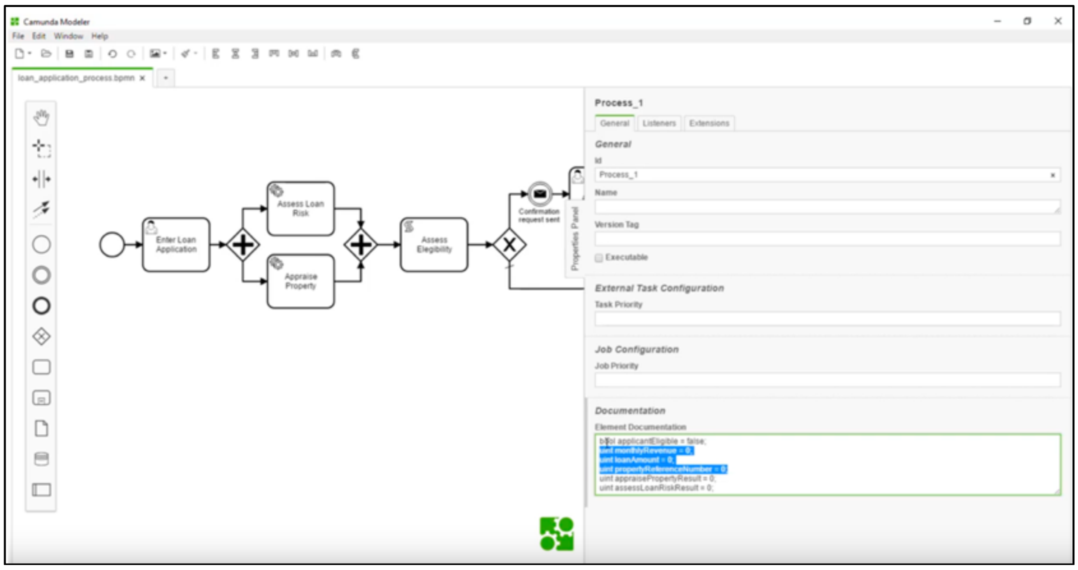
Table 4.5: Sample Contracts against Covered BMPN Elements

Contract Name	BPMN Elements Used
1. Basic Token	Collaboration, Participant, DataOutput, Task, SequenceFlow, IntermediateCatchEvent, Process, MessageFlow, EventBasedGateway, DataInput, DataStoreReference, DataObject, DataStore, StartEvent, EndEvent
2. Lottery	Collaboration, Participant, DataOutput, Task, SequenceFlow, IntermediateCatchEvent, Process, MessageFlow, EventBasedGateway, ExclusiveGateway, ScriptTask, DataObject, DataStore, StartEvent, EndEvent
3. Marriage	Collaboration, Participant, EventBasedGateway, Task, ScriptTask, DataObject, SequenceFlow, StartEvent, EndEvent, Process, MessageFlow

4.3. Analysis of the Available Systems

4.3.1. Caterpillar

Caterpillar [13] [14] is an opensource blockchain based BPMN execution engine. One of the main limitations is the defining meta data such as variable declarations requires annotating 'documentation' attribute of the BPMN Element (please refer Figure 4.11). Due to this limitation, the diagrams drawn using the Caterpillar are abstract and does not encompass the complete image of the contract. In contrast, the proposed approach allows user to define Data Objects which is clearly visible in the BPMN diagram.



Process_1

General | Listeners | Extensions

General

Id: Process_1

Name:

Version Tag:

☐ Executable

External Task Configuration

Task Priority:

Job Configuration

Job Priority:

Documentation

Element Documentation:

```
bool applicantEligible = false;
uint monthlyRevenue = 0;
uint loanAmount = 0;
uint propertyReferenceNumber = 0;
uint appraisePropertyResult = 0;
uint assessLoanRiskResult = 0;
```

Figure 4.10: Annotating Solidity Variables in BPMN Diagram in Caterpillar

In addition, it is apparent that the Caterpillar heavily depends on the '*documentation*' attribute of the BPMN elements and it is overly used for Solidity annotations. In contrast, the proposed approach purely depends on the real essence of the BPM notations and the '*documentation*' attribute is being used for the code documentations.

4.3.2. Unibright

Compared to the proposed approach, Unibright focuses on the work-flow modelling and uses custom notations for the modelling. In contrast, this research avoided introducing entirely new modelling elements. Instead, standard BPMN 2.0 specification is supported in favour of faster community adoption and short learning curve.

Figure 4.12 illustrates an excerpt of the generated code of the workflow example in Figure 2.3(page 27). Complete code generated by Unibright Workflow Designer is available in Appendix A (page 115). Most of the name tags of the approval phases and conditional checks are persisted off the chain. While it improves the efficiency of the resource consumption; it reduces the readability & trust of the on-chain business model.

Compared to the proposed approach which is freely available in Github under Apache2 opensource license, Unibright is a commercial offering, and a high cost is involved and minimal customizations can be done.

.

```
pragma solidity ^0.4.24;

import "./UnibrightContract.sol";

contract MultiPartyApproval is UnibrightContract {

    // the status of the whole approval process
    Status public approvalStatus;

    // holds the timestamp of the last update
    uint public lastUpdate;

    // states the contract and the feedbacks of the approvers can be in
    enum Status { UNDEFINED, NEEDSWORK, APPROVED }

    // uuid of the end node
    bytes16 private endId;

    // map of uuids to approvers, filled during the deployment
    mapping(bytes16 => Approver) private approvers;

    // map of uuids to feedbacks, filled during the deployment
    mapping(bytes16 => Feedback) private feedbacks;

    // map of addresses to uuids, filled during the setup phase
    mapping(address => bytes16) private addresses;

    // Approver holds information about an Approver
    struct Approver {
        Status status;          // the status of the approval
        string comment;         // a description related to the approval status
        uint approvalDate;      // time at which the approver gave his approval
        uint feedbacksNeeded;   // remaining number of feedbacks needed for this
approver to give his approval
        bytes16 feedbackId;     // the id of the feedback object attached to this
approver
        bool exists;            // required to check if mapping exists
    }

    // Feedback holds information about the feedback given by an group of
approvers
    struct Feedback {
        Status status;         // the status of the feedback node
        uint approvalsGiven;   // number of approvals given for this feedback node
        uint approvalsNeeded;  // number of approvals needed for this feedback
node to set its status
        bytes16[] approverIds;// ids of successor approvers attached to the
feedback
        bool exists;            // required to check if mapping exists
    }

    constructor() public {
        approvalStatus = Status.UNDEFINED;
        publish();
        start();
    }

   // ...

    function publish() public onlyBy(owner) {
        super.publish();
        lastUpdate = now;

        addApprover(0xf621af583a18c245b20571c400222fbf,
0x13587eef37d21840a982743534ffdf9a, 0);
        addApprover(0x5484342aa23f9a46aa631e626b3a3480,
0x0ebf120626a889409228a3b88128020e, 1);
        addEndNode(0x98cb347699cccf42be44ad42b55b91d5, 1);
        addApprover(0xd30cbdf0e7ca5f47acb9168910bd8c09,
0x13587eef37d21840a982743534ffdf9a, 0);
        addApprover(0x5b0fe707b73889448efc9c514c9dd881,
0x13587eef37d21840a982743534ffdf9a, 0);

        bytes16[] memory approverList1 = new bytes16[](1);
        approverList1[0] = 0x5484342aa23f9a46aa631e626b3a3480;
        addFeedback(0x13587eef37d21840a982743534ffdf9a, 1, approverList1);
        bytes16[] memory approverList3 = new bytes16[](1);
        approverList3[0] = 0x98cb347699cccf42be44ad42b55b91d5;
        addFeedback(0x0ebf120626a889409228a3b88128020e, 1, approverList3);
    }
   // ...
}
```

Figure 4.12: Unibright excerpt code for the multi-party approval template example (complete code is available in Appendix A, page 115)

4.3.3. Petri-Nets Translation and Reduction Approach

Compared to the proposed approach, BPMN to the Petri-Nets transformation, is supporting only a limited subset of BPMN elements. An excerpt of the generated solidity code derived from [16] is shown in Figure 4.13. As per the figure, traceability of the BPMN core elements in generated code is comparatively lower than the proposed approach due to additional overhead added to the contracts. However, the proposed approach contains minimum overhead and the possible optimizations are out of scope for our research.

```
contract BPMNContract {
  uint marking = 1;
  uint predicates = 0;
  function CheckApplication( – input params – ) returns (bool) {
    if (marking & 2 == 2) { // is there a token in place p1 ?
      // Task B's script goes here, e.g. copy value of input params to contract variables
      uint tmpPreds = 0;
      if ( – evalP – ) tmpPreds |= 1; // is loan application complete?
      if ( – evalQ – ) tmpPreds |= 2; // is the property pledged?
      step(
        marking & uint(~2) | 12,          // New marking
        predicates & uint(~3) | tmpPreds // New evaluation for "predicates"
      );
      return true;
    }
    return false;
  }
  function AppraiseProperty(uint tmpMarking) internal returns (uint) {
    // Task E's script goes here
    return tmpMarking & uint(~8) | 32;
  }
  function step(uint tmpMarking, uint tmpPredicates) internal {
    if (tmpMarking == 0) { marking = 0; return; } // Reached a process end event!
    bool done = false;
    while (!done) {
        // does p3 have a token and does P ∧ Q hold?
        if (tmpMarking & 8 == 8 && tmpPredicates & 3 == 3) {
          tmpMarking = AppraiseProperty(tmpMarking);
          continue;
        }
        // does p3 have a token and does P ∧ ¬Q hold?
        if (tmpMarking & 8 == 8 && tmpPredicates & 3 == 2) {
          tmpMarking = tmpMarking & uint(~8) | 32;
          continue;
        }
        ...
        done = true;
    }
    marking = tmpMarking; predicates = tmpPredicates;
  } ... }
```

Figure 4.13: An Excerpt of the Generated Solidity Code in [16]

4.3.3. Choreography Diagrams Approach

In this research, BPMN 2.0 choreography diagrams to model business processes targeting blockchains. The Solidity code generation process algorithm for the business contracts is listed in Figure 4.14.

```
Bool $Task_1Activated \leftarrow false$
...
Bool $Task_nActivated \leftarrow false$

Bool $JoinGateway_1Incoming_1Activated \leftarrow false$
...
Bool $JoinGateway_1Incoming_nActivated \leftarrow false$
Bool $JoinGateway_nIncoming_1Activated \leftarrow false$
...
Bool $JoinGateway_nIncoming_nActivated \leftarrow false$

Bool TerminationActivated $\leftarrow false$

function INIT()
    $Task_1Activated \leftarrow true$
end function

function TASK$_1$()
    ...
end function
...
function TASK$_n$()
    ...
$TerminationActivated \leftarrow true$
end function

function JOINGATEWAY$_1$()
    ...
end function
...
function JOINGATEWAY$_n$()
    ...
end function
```

Figure 4.14: Process Instance Contract Implementation Templating Algorithm [18]

Opposed to the proposed approach, this approach has considered only a limited set of BPMN elements including '*Task*', '*Sequence Flow*', '*Start Event*', '*End Event*', '*Parallel Gateway*' and '*Data-based XOR Gateway*' which claimed to be sufficiently covers the basic control-flow patterns of the process modelling [17]. Nevertheless, our goal was to cover BPMN elements of the standard specification as much as possible in order to minimize the drawing effort and make the diagram more intuitive. An extensive gap analysis is done in section 7.4.7 (page 92) to identify which elements are to be supported as the future directives.

4.3.3. Extended Choreography Diagrams Approach

For this research, suggested an extended version of the BPMN 2.0 choreography diagrams [7] to generate the Solidity code. An excerpt generated code of a common interface can be found in Figure 4.15. The complete code for the Rental Agreement example can be found in their online appendix [11].

```
pragma solidity ^0.4.23;
interface Choreography { [...]
    function start() external;
    function sendRequest(uint8 task, bytes message) external;
    function sendResponse(uint8 task, bytes message) external;
}
```

Figure 4.15: Code of the Generated Common Interface (excerpt)

Compared to the proposed approach, current research avoided introducing entirely new modelling elements and extensions to the standard BPMN 2.0 specification. Instead, available BPMN 2.0 process elements were opted to reuse. On the other hand, choreography diagrams by nature, only explains the messaging between multiple parties but not their internal states. In contrast, BPMN 2.0 process model & collaborations diagrams in our approach represents the entire communication between multiple parties and their state changes.

The below Table 4.6 compares the main features of the similar translators.

[11] Generated code for the rental agreement choreography diagram example can be found in https://github.com/jan-ladleif/bpm19-blockchain-based-choreographies/tree/5785682668176fe931338e50c3df01b3518faf5c/contracts/rental-agreement online, accessed 26-11-2019

Table 4.6: Summary of the Similar Translators

	Approach	BPMN 2.0 Support	Backword Traceability of modelling elements in code	Diagram support for the complex transactions	Public availability of the Implementation	License
Unibright [15]	A template based, visual definition of business integration workflows	No, uses a custom set of notations provided by templates	Comparatively LOW. Templates based. Uses off-chain meta data to maintain name tags.	Comparatively LOW. Modelling elements available are not sufficient*.	No, only a try-out demo is available	Commercial
Caterpillar [13]	Assumes a Solidity annotated BPMN diagram	Yes	Comparatively LOW. Templates based.	Comparatively LOW. Solidity annotations are required for the diagram	Yes, in Github[13]	BSD 3-Clause License
BPMN and PetriNets reductions [16]	BPMN transformed into PetriNets. After reductions, transformed into Solidity.	Yes	Comparatively LOW. Additional overhead is added to the contracts.	Comparatively LOW. Petri net supports only a limited subset of BPMN elements	No	N/A
Choreography diagrams [17]	BPMN 2.0 choreography diagrams to	Yes	Comparatively LOW.	Comparatively LOW. Only limited set of BPMN	No	N/A

[13] Caterpillar implementation is available in https://github.com/orlenyslp/Caterpillar online, accessed 18-11-2019

	Approach	**BPMN 2.0 Support**	**Backword Traceability of modelling elements in code**	**Diagram support for the complex transactions**	**Public availability of the Implementation**	**License**
	model business processes			elements supported		
Extended Choreography diagrams [7]	An extended version of the BPMN 2.0 choreography diagrams	No, Extended version of BPMN choreography diagrams used	Comparatively LOW.	Similar, Provides similar set of BPMN elements through BPMN extensions	No	N/A
Proposed Approach	Uses code generation approach via token-based AST translation	Yes	Comparatively HIGH due to well defined one-to-one BPMN-to-Solidity mapping	Comparatively HIGH** due to extensive coverage of the BPMN elements.	Yes, In Github[14]	Apache2

*Only available public template for the Unibright is multi approval template
** Similar only to the extended choreography diagrams in [7] which provides similar set of elements.

[14] Proposed approach's implementation is available in https://github.com/rasika/bpmn-to-solidity online, accessed 08-12-2019

4.3.3. Comparative Analysis of the Diagrams

A process modelling diagram and the generated Solidity code for a generic example (i.e. Loan assessment process) is compared against the Caterpillar system [13] [14] and the Proposed approach.

It is worth to note that only Caterpillar and Unibright implementations are available to the public. However, the Unibright does not allow to draw the complete diagram, as the limited nodes are freely available (merely the approval process is available). Therefore, a complete diagram for the loan assessment process could not been drawn. Therefore, the only the Caterpillar was evaluated against the proposed approach.

Example used is derived from the Caterpillar v1.0 implementation[11], which is a simplified loan assessment process that accepts 'cost', 'monthly revenue' and 'loan amount' as inputs. Loan risk and property appraisal verification is done via external API calls for the Oracle nodes. Upon both successes results from the oracle nodes, in order to accept the loan application, the final confirmation need to be sent.

Caterpillar v3.0

The diagram from Caterpillar v1.0 has been slightly modified to work with Caterpillar v3.0. Changes are done for the *document* attribute of the BPMN elements using Solidity annotations specification. The BPMN XML for the modified diagram is available in Appendix B: B1 (page 121). Figure 4.16 shows the process model diagram.

[11] The example source BPMN file of the Caterpillar is available in https://github.com/orlenyslp/Caterpillar/blob/f4789f2a708627135e55eb972a0f3c9ed8edcc71/v1.0/caterpillar-core/demo_running_example.bpmn online, accessed 02-12-2019

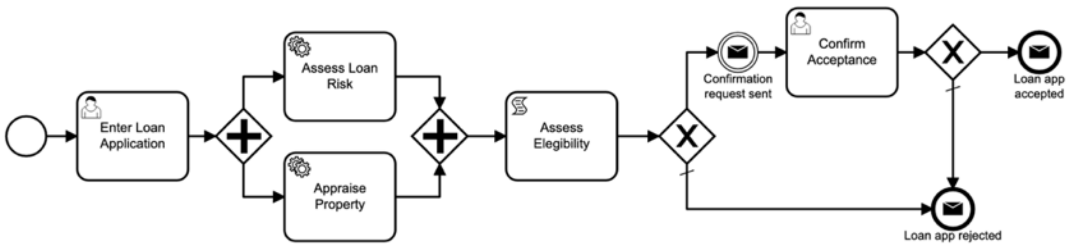


Figure 4.16: BPMN Diagram of the Loan Assessment Process Model using Caterpillar [13]

In here, few state level variables have been defined in *documentation* of the global process level (refer Figure 4.17).

```
<bpmn:process id="BPM17_Running_Example" name="BPM17_Running_Example" isExecutable=
"false">
    <bpmn:documentation>
<![CDATA[bool applicantEligible;
uint monthlyRevenue;
uint loanAmount;
uint cost;
uint appraisePropertyResult;
uint assessLoanRiskResult;
]]>
    </bpmn:documentation>
</bpmn:process>
```

Figure 4.17: Solidity Annotations Required for the Global Process Level in Caterpillar

Tasks need to be annotated with Solidity annotations. For instance, 'Enter Loan Application' task contains the below Solidity annotations (refer Figure 4.18);

```
<bpmn:userTask id="Task_06xlgcp" name="Enter Loan Application">
    <bpmn:documentation>
<![CDATA[@ Manager @
() : (uint _monthlyRevenue, uint _loanAmount, uint _cost) -
> monthlyRevenue = _monthlyRevenue; loanAmount = _loanAmount; cost = _cost;]]>
    </bpmn:documentation>
</bpmn:userTask>
```

[<@ role_name @>] (data_to_export) : (data_to_import) ->
{Operations_to_perform}

Figure 4.18: Solidity Annotations Required for the 'Enter Loan Application' Task in Caterpillar

Solidity annotation syntax for the BPMN documentation element is shown in Figure 4.19.

[<@ role_name @>] (data_to_export) : (data_to_import) ->
{Operations_to_perform}

Figure 4.19: Solidity Annotation Syntax for the Tasks in Caterpillar [13]

Proposed Approach

BPMN diagram using the proposed approach is portrayed in Figure 4.20. Inputs for the diagram is clearly visible and persistence of the variables are in BPMN elements rather than Solidity annotations. The diagram of the proposed approach is more verbose and provides the complete picture and solely compliant with the BPMN 2.0 specification. The BPMN XML code for the diagram can be found in Appendix B, B2 (page 128).

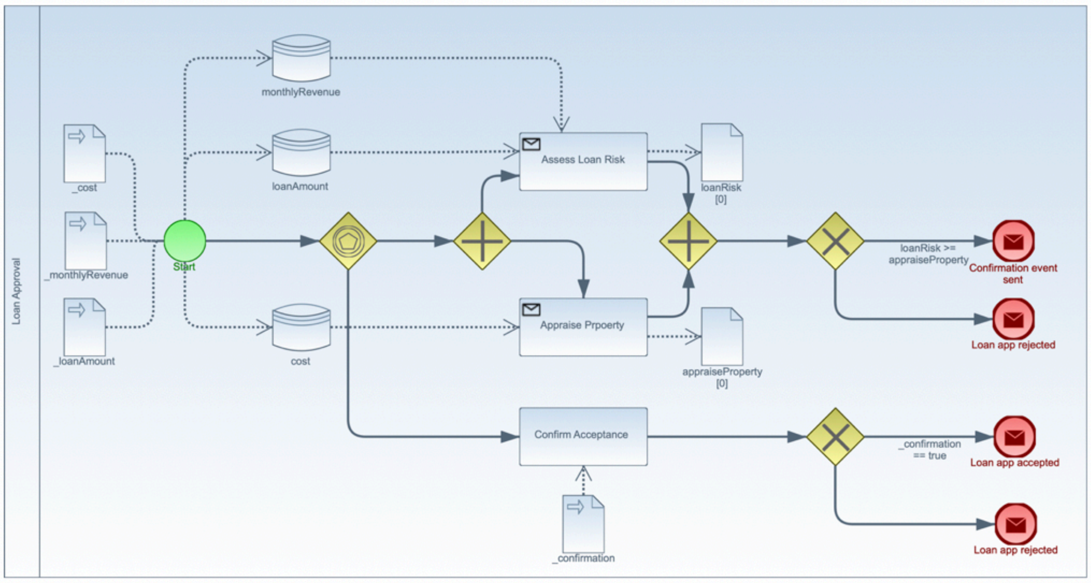
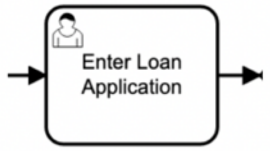
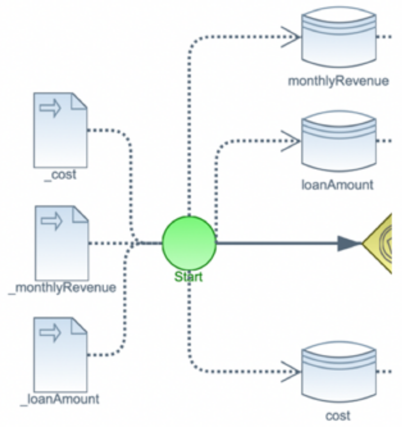


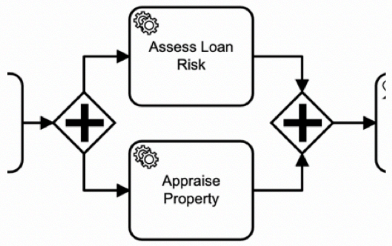
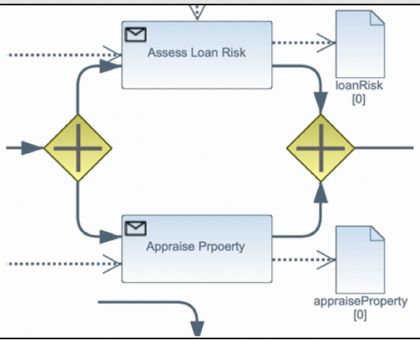
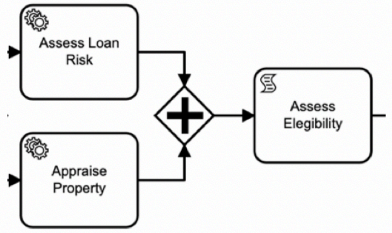
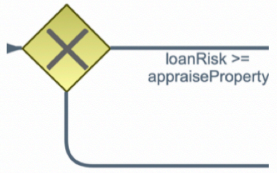
Figure 4.20: BPMN Diagram of the Loan Assessment Process using the Proposed Approach

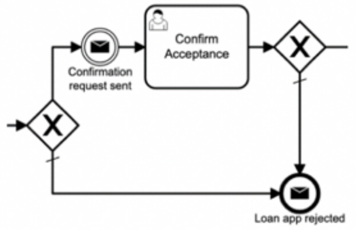
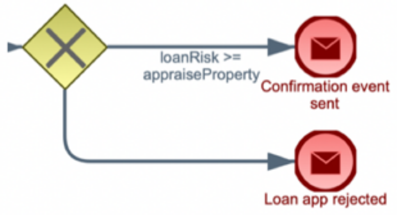
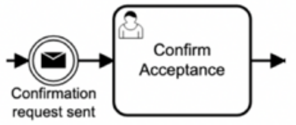
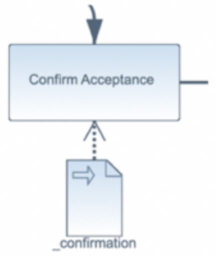
Table 4.7 compares BPMN representations between the Caterpillar and the Proposed approach step-by-step using the loan assessment example.

.

Table 4.7: Loan Assessment Steps Comparison

Process Model Step	BPMN Representation in Caterpillar	BPMN Representation in Proposed Approach
(1) A loan application contract is started with providing 'loan amount', 'monthly revenue' and the 'cost' of the institution.	A **UserTask** with the '**documentation'** attribute containing variable definitions; 'loan amount', 'monthly revenue' and 'cost'. 	A **StartEvent** with **DataInputs** and associated **DataObjects** for the variables 'loan amount', 'monthly revenue' and 'cost'(A StartEvent with inputs represents a parameterized Solidity Constructor.)  monthlyRevenue loanAmount cost _cost _monthlyRevenue _loanAmount Start
(2) Need to evaluate the 'loan risk' and 'appraise the property provided' using external API calls off-the-chain.	Two **ServiceTasks** combined by **ParallelGateways** for the 'risk assessment' and 'property appraisal'. External API calls and outputs of the oracle nodes are mapped into two local variables defined in '**documentation'** attribute of each ServiceTask. *Mapped variables are not visible in the graphical diagram.	Two **ReceiveTasks** combined by **ParallelGateways** for the 'risk assessment' and 'property appraisal'. External API calls are defined in the interface details of the respective **ReceiveTasks**. Outputs of the oracle nodes are mapped into two local variables using **DataObjects** 'loanRisk' and 'appraiseProperty'.

Process Model Step	BPMN Representation in Caterpillar	BPMN Representation in Proposed Approach
		 Assess Loan Risk loanRisk [0] Appraise Prpoerty appraiseProperty [0]
(3) Assess the eligibility. The Eligibility meets when risk of the loan is less than or equal to the cost appraisal.	A **ScriptTask** contains the comparison logic. 	An **ExclusiveGateway** with a conditional **SequenceFlow** is used. The comparison logic is clearly visible to the modeler. 

Process Model Step	BPMN Representation in Caterpillar	BPMN Representation in Proposed Approach
(4) If eligible, send confirmation event notification. Or else, send loan application rejected notification.	ConfirmationRequest **MessageEvent** sends the notification.  Confirmation request sent Confirm Acceptance Loan app rejected	**EndEvent** with a **MessageEvent** sends the notification.  loanRisk >= appraiseProperty Confirmation event sent Loan app rejected
(5) Loan application approved notification is sent upon confirmation. Or else, send loan application rejected notification.	A **UserTask** with the **'documentation'** attribute containing variable '_confirmation' retrieves the input. *The input variable is not visible through the diagram. 	A **UserTask** with **DataInput** '_confirmation' retrieves the input. 

4.3.3. Comparative Analysis of the Generated Solidity Codes

Comparative analysis of the generated Solidity code is twofold.

1. Static Analysis of the Generated Solidity Source Code
2. Blockchain Deployment Analysis of the Generated Code

Static Analysis of the Generated Solidity Source Code

Hegedűs P. [21] has conducted a research to analyse the code complexity of Solidity based smart contracts. The proposed source code metrics for the Solidity language are listed in Table 4.8.

Table 4.8: Solidity Source Code Metrics for Smart Contracts [21]

Metric			Description	Explanation
1.	**SLOC**	Source Lines of Code	Number of source code lines	Lower the better. When SLOC increases the readability is decreased.
2.	**LLOC**	Logic Lines of Code	Number of logical code lines (empty and comment lines are ignored)	Lower the better. When LLOC increases the complexity is increased, thus lower the readability.
3.	**CLOC**	Comment Lines of Code	Number of comment lines	Higher the better. When CLOC increases the readability is increased.
4.	**NF**	Number of Functions	Number of functions	Lower the better. When NF increases, lowers the traceability of the human.
5.	**WMC**	Weighted McCabe's Complexity	The sum of McCabe's style complexity over the functions of a contract	Lower the better. When WMC increases the complexity is increased
6.	**NL**	Nesting Level	The sum of the deepest nesting level of the control structures	Lower the better, When NL increases the complexity is increased
7.	**NLE**	Nesting Level else-if	Number of nesting level else-if	Lower the better, When NLE increases the complexity is increased

8.	**NUMPAR**	Number of Parameters	Number of parameters over the functions	Lower the better, When NUMPAR increases the complexity is increased
9.	**NOS**	Number of Statements	Number of statements	Lower the better, When NOS increases the complexity is increased
10.	**NOA**	Number of Ancestors	Number of different direct or transitive ancestors	Lower the better, When NOA increases the complexity is increased
11.	**NA**	Number of Attributes	Number of state variables	Lower the better, When NA increases the complexity is increased
12.	**NOI**	Number of Outgoing Invocations	Number of different function invocations within functions	Lower the better, When NOI increases the complexity is increased

In order to calculate these metrics, the author has implemented a prototype tool called SolMet[11]. SolMet is able to parse Solidity source files and calculate source code metrics listed in above Table 4.8. Accordingly, same tool has been used to analyze the generated Solidity source code for the loan assessment example against publicly available BPMN-to-Solidity translator system (i.e. Caterpillar).

Results from the SolMet tool against the generated source code is depicted in Table 4.10. According to the metrics results, there is a significant gap between the numbers. The reason for this difference is that Caterpillar generates contract templates which has substantial amount of lines of code and logical lines.

For instance, in order to deploy the loan assessment example, Caterpillar first requires a process registry (114 lines). Then the drawn process model can be deployed. For the process model, Caterpillar deploys a BPMN Interpreter (341 lines), an IFlow template (147 lines) and a Factory template (7 lines). For the Caterpillar, all required contract source files of the loan assessment example are listed in Table 4.10. The primitive way of comparing the complexity is the lines of code. Accordingly, *Caterpillar* has

[11] The SolMet tool is available on https://github.com/chicxurug/SolMet-Solidity-parser online, accessed 17-12-2019

a total of **762 lines** of code. In contrast, the *Proposed Approach* only requires a single file of **58 lines** of code (refer Appendix C, page 137).

When analyzing the metrics, it is apparent that the average SLOC, LLOC, CLOC per contract are higher compared to the proposed approach. Higher SLOC and LLOC increases complexity of the code. However, CLOC is comparatively low in the proposed approach and it is satisfactory since low number of lines appear in the code. The number of functions (denoted as NF) for the Caterpillar is 104 and average of 10.4 per contract. In contrast, proposed approach has two (2), namely '*confirmAcceptance*' and '*__callback*'. Moreover, as per the [22], WMC for the proposed approach is rated as 'Low' level risk, compared to the Caterpillar WMC for individual source files is between 'Moderate', 'High' and 'Very High' risk levels.

Table 4.9: Cyclomatic Complexity Recommendations [22]

CC	Type of Procedure	Risk
1 to 4	A simple procedure	Low
5 to 10	A well-structured and stable procedure	Low
11 to 20	A more complex procedure	Moderate
21 to 50	A complex procedure, worrisome	High
> 50	An error-prone, extremely troublesome, untestable procedure	Very high

Further, the deepest nesting level (denoted as NL) and else-if nesting levels (denoted as NLE) for the Caterpillar is comparatively high. Number of parameters (denoted as NUMPAR) and number of state variables (denoted as NOS) is also comparatively high. Increase of these numbers believed to be increasing the complexity of the code [22]. Number of direct and transitive ancestors are same for the both approaches. Moreover, average number of state variables (denoted as NA) and average number of function invocations in function definitions (denoted as NOI) are also higher in Caterpillar compared to proposed approach.

.

Table 4.10: Generated Solidity Code Complexity Analysis Metrics Results

System	Solidity Files	Contract Names (N)	SLOC	LLOC	CLOC	NF	WMC	NL	NLE	NUMPAR	NOS	NOA	NA	NOI
Cv3	IFlow.sol	IFlowImpl	121	81	9	18	24	5	5	21	34	0	11	4
Cv3		IFlow	26	19	3	18	18	0	0	21	0	0	0	0
Cv3	ProcessRegistry.sol	ProcessRegistry	105	83	0	15	19	4	4	24	42	0	8	14
Cv3		IFunct	9	7	2	5	5	0	0	5	0	0	0	0
Cv3	IData.sol	IInterpreter	4	3	0	2	2	0	0	2	0	0	0	0
Cv3		IData	18	17	0	15	15	0	0	14	0	0	0	0
Cv3		IDataImp	73	53	0	14	14	0	0	13	18	0	7	4
Cv3	BPM17_Running_ ExampleFactory.sol	BPM17_Running_ ExampleFactory	7	3	0	1	1	0	0	0	1	0	0	1
Cv3	BPMNInterpreter.sol	BPMNInterpreter	341	206	78	10	57	25	19	18	164	0	0	87
Cv3	IFactory.sol	IFactory	3	2	0	1	1	0	0	0	0	0	0	0
Cv3	BPM17_Running_ ExampleData.sol	BPM17_Running_ ExampleData	55	46	0	5	13	4	3	10	26	1	6	7
TOTAL			762	520	92	104	169	38	31	128	285	1	32	117
AVERAGE (TOTAL / N = 10)			76.2	52	9.2	10.4	16.9	3.8	3.1	12.8	28.5	0.1	3.2	11.7
PrA	LoanApproval.sol	LoanApproval	58	44	2	2	5	2	2	3	22	1	6	2
TOTAL			58	44	2	2	5	2	2	3	22	1	6	2
AVERAGE (TOTAL / N = 1)			58	44	2	2	5	2	2	3	22	1	6	2

N = Number of Contracts

Cv3 = Caterpillar v3.0, PrA = Proposed Approach

Figure 4.21 depicts the summary of metrics results of the total generated Solidity code complexity.

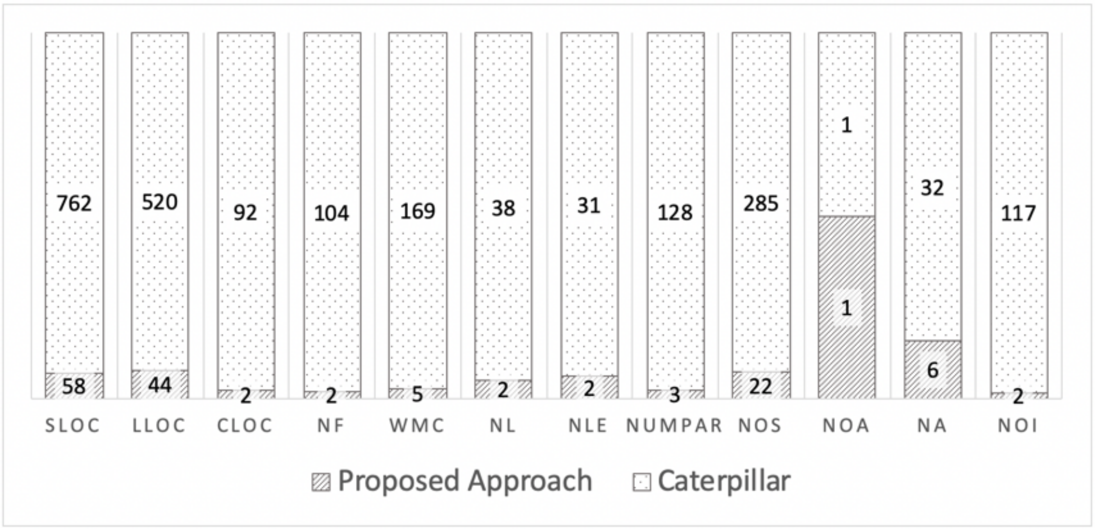


Figure 4.21: Summary of Metrics Results of the Total Solidity Contract Complexity

Figure 4.22 depicts the summary of metrics results of the average Solidity code complexity per contract. Average values are derived from the results of the Figure 4.21, divided by the number of contracts(N) in total.

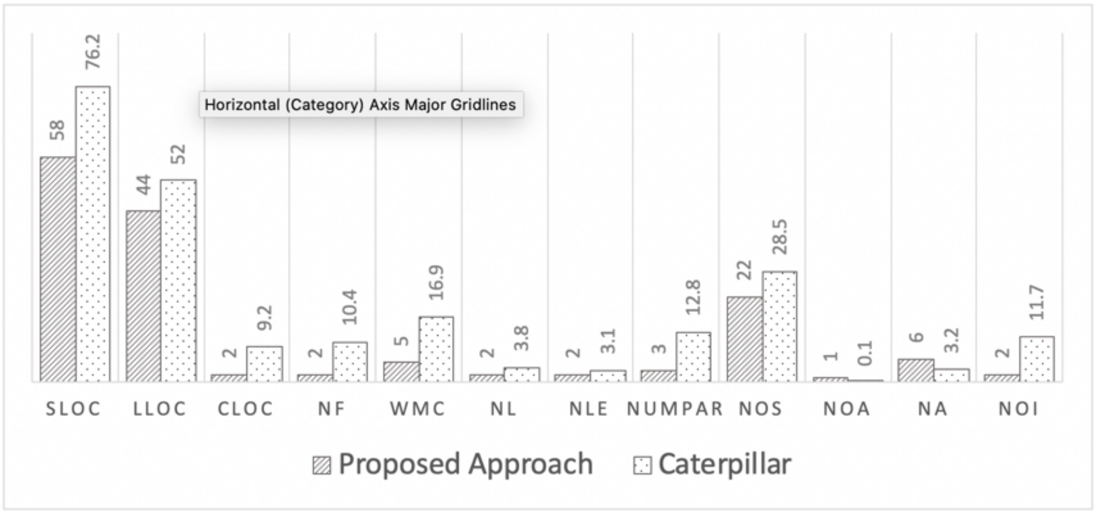
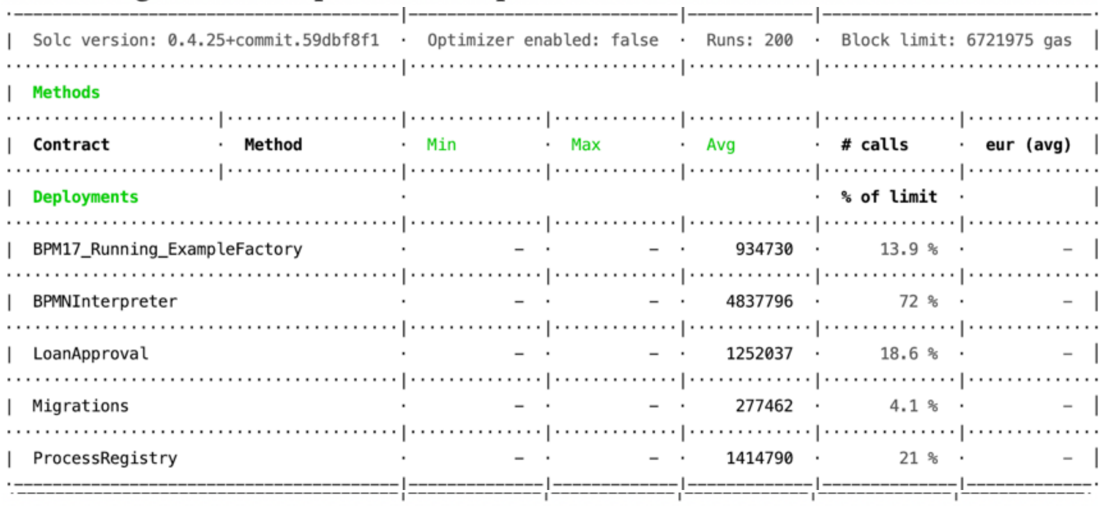


Figure 4.22: Summary of the Average Metrics Results of Solidity Contract Complexity

In overall, results demonstrate that the Caterpillar generates a substantial amount of lines of code (762 lines) for the loan assessment example whereas the proposed approach generates a minimum number of lines of code (58 lines). Other metrics, prove that the complexity of the total generated code and the average complexity per contract is comparatively high for the Caterpillar.

Blockchain Deployment Analysis of the Generated Solidity Code

Every operation made on Ethereum, requires gas as the execution fee[11]. In order to analyse the gas consumption and the deployment ability without any errors, the generated code from the Caterpillar and the proposed approach has been deployed in a blockchain test network using Truffle[12] test framework for Solidity. In order to retrieve gas details, a JavaScript plugin; eth-gas-reporter[13] installed and configured into truffle-config file. Below Figure 4.23 depicts the output of the truffle test framework.



```
·--------------------------------------|-------------------------|-----------|-------------------------·
|  Solc version: 0.4.25+commit.59dbf8f1  ·  Optimizer enabled: false  ·  Runs: 200  ·  Block limit: 6721975 gas  |
·······································|·························|···········|··························
|  Methods                                                                                              |
·······················|···············|·············|···········|···········|·············|···········
|  Contract            ·  Method       ·  Min        ·  Max      ·  Avg      ·  # calls    ·  eur (avg)  |
·······················|···············|·············|···········|···········|·············|···········
|  Deployments                         ·                                     ·  % of limit  ·             |
·······································|·············|···········|···········|·············|···········
|  BPM17_Running_ExampleFactory        ·          -  ·        -  ·   934730  ·     13.9 %  ·          -  |
·······································|·············|···········|···········|·············|···········
|  BPMNInterpreter                     ·          -  ·        -  ·  4837796  ·       72 %  ·          -  |
·······································|·············|···········|···········|·············|···········
|  LoanApproval                        ·          -  ·        -  ·  1252037  ·     18.6 %  ·          -  |
·······································|·············|···········|···········|·············|···········
|  Migrations                          ·          -  ·        -  ·   277462  ·      4.1 %  ·          -  |
·······································|·············|···········|···········|·············|···········
|  ProcessRegistry                     ·          -  ·        -  ·  1414790  ·       21 %  ·          -  |
·--------------------------------------|-------------|-----------|-----------|-------------|-----------·
```

Figure 4.23: Truffle Framework Output of the Deployment of Contracts

Below Table 4.11 derived from output of the truffle framework in Figure 4.23. Solidity code generation time is calculated by adding console logs to the Caterpillar source code and the proposed approach required no changes since it is already logging time consumed.

Table 4.11: Deployment Gas consumption Identification

System	Contracts Generated	Generation Time	Gas Consumed	Total Gas Consumed
Caterpillar	Process Registry		1,414,790	
	BPMN Interpreter		4,837,796	
	BPM17_Running_ExampleFactory	**~20ms**	934,730	**7,187,316**
Proposed Approach	Loan Approval	**~263ms**	1,252,037	**1,252,037**

According to the Table 4.11, Solidity source code generation time for the proposed approach is comparatively high compared to Caterpillar. One reason for this delay is due to Java implementation of the PoC, whereas the Caterpillar was implemented as a web application using TypeScript. Further, in terms of total gas consumption, Caterpillar requires a substantial amount of gas compared to the Proposed System. This is mainly due to high number of lines of code.

4.4. Input BPMN Diagrams and Re-generated Solidity Codes

This section demonstrates the completed BPMN diagrams for different business use cases such as; Rental agreement, Marriage: wedding gift, Sellable, Lottery, Basic token and Crowd sale, which were drawn using BPMN2.0 modeler and their respective re-generated Solidity codes using the proposed approach.

4.4.1. Rental Agreement

Rental agreement is a contract between a landlord and tenant. The purpose of the contract is to make sure that rent is paid on time. This contract is obtained from [23].

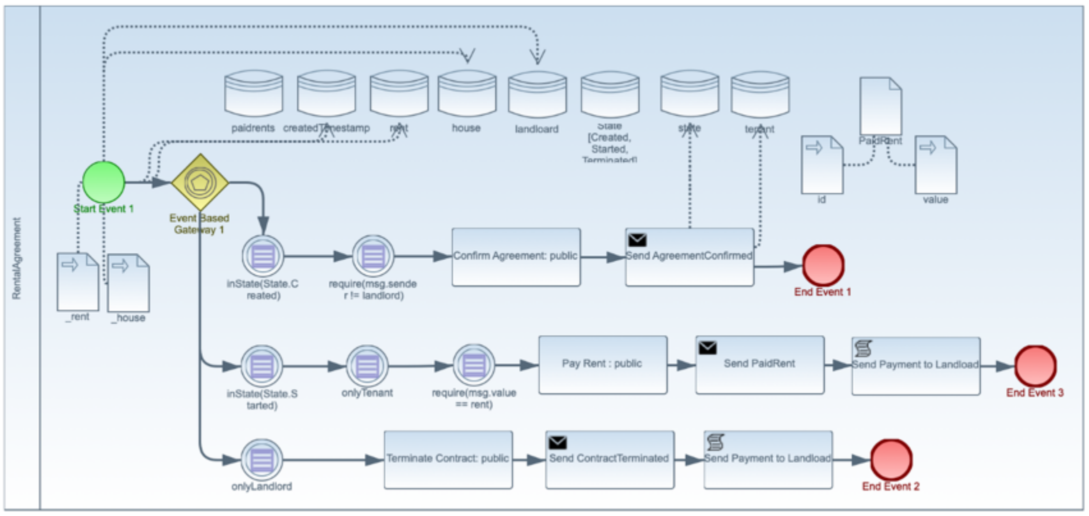


Figure 4.24: BPMN Diagram for the Rental Agreement Business Contract

```
pragma solidity >=0.4.0 <0.7.0;

contract RentalAgreement {

    struct PaidRent {
        uint id;
        uint value;
    }

    PaidRent public paidrents;
    uint public createdTimestamp;
    uint public rent;
    string public house;
    address public landloard;
    State public state;
    address public tenant;
    enum public State {Created, Started, Terminated};

    event AgreementConfirmed();
    event ContractTerminated();
    event PaidRent();

    constructor (uint _rent, string _house){
        rent = _rent;
        house = _house;
        landloard = msg.sender;
        createdTimestamp = block.timestamp;
    }

    modifier inState(State.Created) {
        if (state == _state) _;
    }

    modifier onlyLandlord {
        if (msg.sender != tenant) _;
    }

    modifier inState(State.Started) {
        if (state == _state) _;
    }

    modifier onlyTenant {
        if (msg.sender == tenant) _
    }

    /** Confirm the lease agreement as tenant */
    function confirmAgreement() public inState(State.Created) require(msg.sender !=
landlord) {
        emit AgreementConfirmed();
    }

    /** Terminate the contract so the tenant can't pay rent anymore, and the contract is
terminated */
    function terminateContract() public onlyLandlord {
        emit ContractTerminated();
        landlord.send(this.balance);
        state = State.Terminated;
    }

    function payRent() public inState(State.Started) onlyTenant require(msg.value == rent)
{
        emit PaidRent();
        landlord.send(msg.value);
        paidrents.push(PaidRent({
        id : paidrents.length + 1,
        value : msg.value
        }));
    }
}
```

Figure 4.25: Re-generated Solidity Code for the Rental Agreement Business Contract

4.3.3. Marriage: Wedding Gift

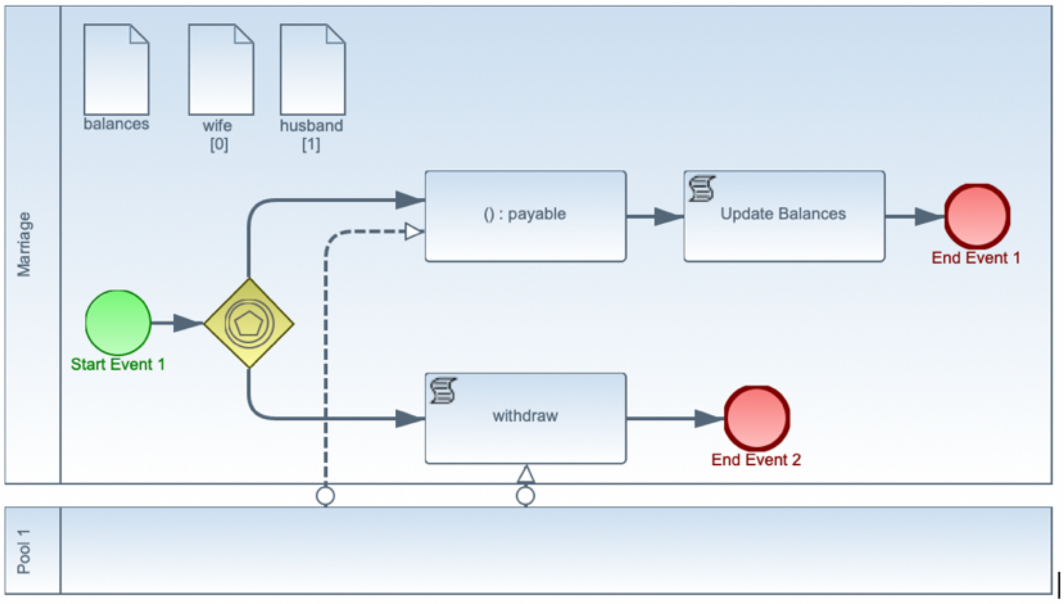


Figure 4.26: BPMN Diagram for the Marriage_Wedding Gift Business Contract

```
pragma solidity >=0.4.0 <0.7.0;

contract Marriage {
    mapping (address => uint) balances;
    address wife = address(0);
    address husband = address(1);

    function withdraw () {
        uint amount = balances[msg.sender];
        balances[msg.sender] = 0;
        msg.sender.transfer(amount);
    }
    function () payable {
        balances[wife] += msg.value / 2;
        balances[husband] += msg.value / 2;
    }
}
```

Figure 4.27: Re-generated Solidity Code for the Marriage_Wedding Gift Business Contract

4.3.4. Sellable

In this contract there are three participants; owner, buyer and the contract. Therefore, this contract elaborates a sales transaction occur among these three parties. This contract is obtained from [24].

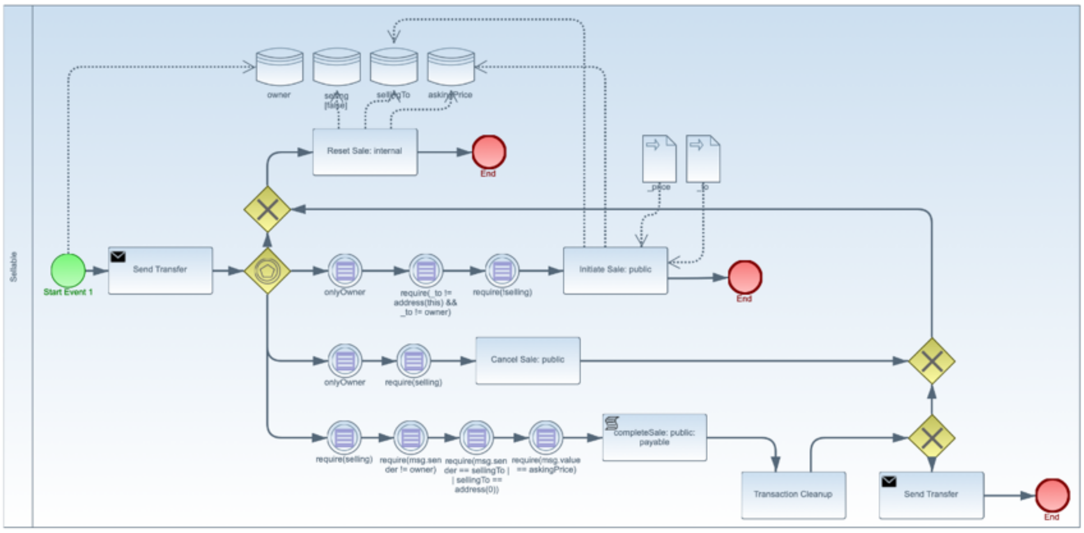


Figure 4.28: BPMN Diagram for the Sellable Business Contract

```
pragma solidity >=0.4.0 <0.7.0;

contract Sellable {

    // The owner of the contract
    address public owner;

    // Current sale status
    bool public selling = false;

    // Who is the selected buyer, if any.
    address public sellingTo;

    // How much ether (wei) the seller has asked the buyer to send
    uint public askingPrice;

    event Transfer(uint _saleDate, address _from, address _to, uint _salePrice);

    constructor (){
        owner = msg.sender;
        emit Transfer(_saleDate, _from, _to, _salePrice);
    }

    modifier onlyOwner {
        require(msg.sender == owner);
        _;
    }

    function initiateSale(uint _price, address _to) public onlyOwner {
        require(_to != address(this) && _to != owner);
        require(!selling);
        selling = true;
        sellingTo = _to;
        askingPrice = _price;
    }

    function cancelSale() onlyOwner public {
        require(selling);
        resetSale();
    }

   function completeSale() public payable {
        require(selling);
        require(msg.sender != owner);
        require(msg.sender == sellingTo || sellingTo == address(0));
        require(msg.value == askingPrice);

        address prevOwner = owner;
        address newOwner = msg.sender;
        uint salePrice = askingPrice;

        owner = newOwner;
        resetSale();
        prevOwner.transfer(salePrice);

        Transfer(now,prevOwner,newOwner,salePrice);
    }

    function resetSale() internal onlyOwner {
        selling = false;
        sellingTo = address(0);
        askingPrice = 0;
    }
}
```
Figure 4.29: Re-generated Solidity Code for the Sellable Business Contract

4.3.3. Lottery

The business logic of the lottery contract is explained below. This contract is obtained from [25].

- Limit is 10 users
- User has to pay 0.1 ether to join the lottery
- Same user can join once
- Owner of the contract can join the lottery
- When 10 users join then the winner is picked
- Winner receives all the money
- New lottery starts when the winner is picked

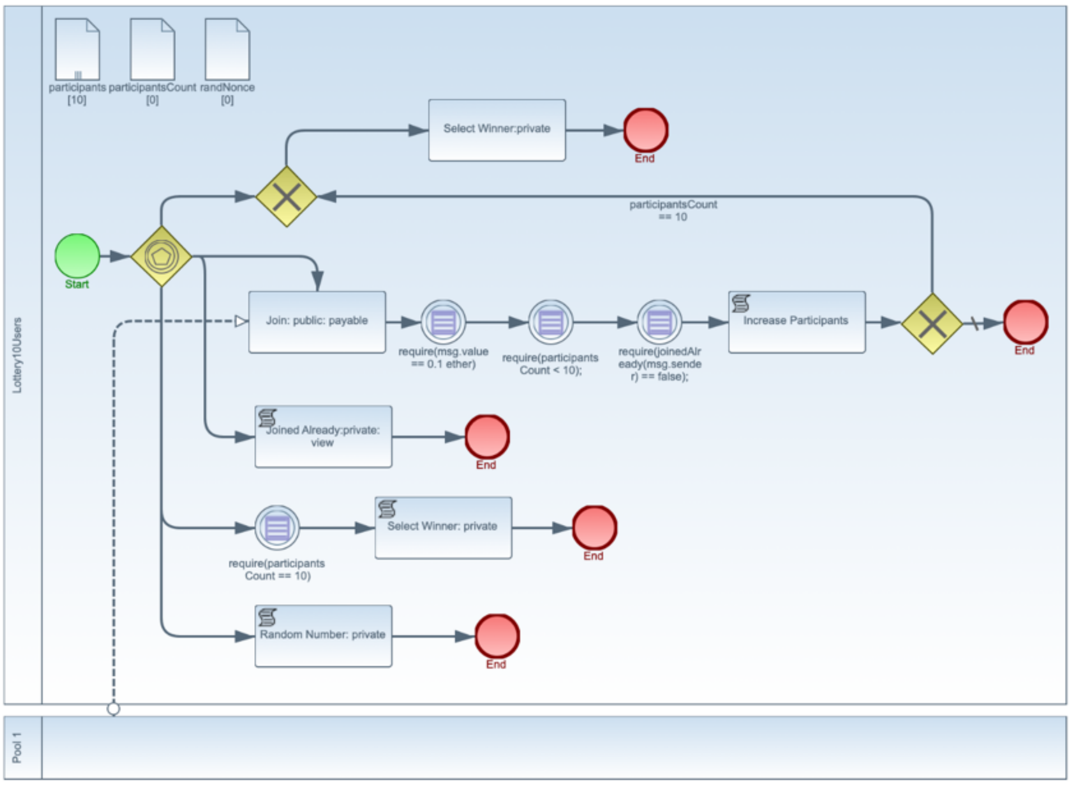


Figure 4.30: BPMN Diagram for the Lottery Business Contract

```
pragma solidity >=0.4.0 <0.7.0;

contract Lottery10Users {
    address[10] participants;
    uint8 participantsCount = 0;
    uint randNonce = 0;

    function join() public payable {
        require(msg.value == 0.1 ether, "Must send 0.1 ether");
        require(participantsCount < 10, "User limit reached");
        require(joinedAlready(msg.sender) == false, "User already joined");
        participants[participantsCount] = msg.sender;
        participantsCount++;
        if (participantsCount == 10) {
            selectWinner();
        }
    }

    function joinedAlready(address _participant) private view returns(bool) {
        bool containsParticipant = false;
        for(uint i = 0; i < 10; i++) {
            if (participants[i] == _participant) {
                containsParticipant = true;
            }
        }
        return containsParticipant;
    }

    function selectWinner() private returns(address) {
        require(participantsCount == 10, "Waiting for more users");
        address winner = participants[randomNumber()];
        winner.transfer(address(this).balance);
        delete participants;
        participantsCount = 0;
        return winner;
    }

    function randomNumber() private returns(uint) {
        uint rand = uint(keccak256(abi.encodePacked(now, msg.sender, randNonce))) % 10;
        randNonce++;
        return rand;
    }

}
```
Figure 4.31: Re-generated Solidity Code for the Lottery Business Contract

4.3.3. Basic Token

The business logic of the Basic Token contract is explained below. This contract is obtained from [25].

- Initial supply of tokens is set on creation
- Contract creator gets initial tokens
- Tokens can be transferred to any account
- There is a protection from overflow
- Everyone can check balances

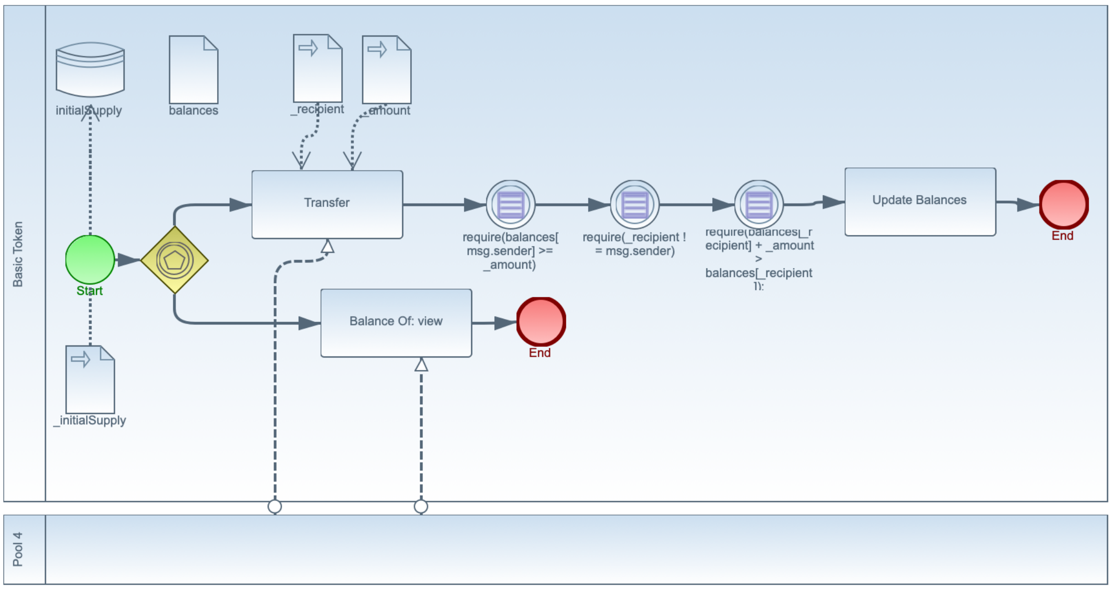

Basic Token
initialSupply
balances
_recipient
_amount
Transfer
require(balances[msg.sender] >= _amount)
require(_recipient != msg.sender)
require(balances[_recipient] + _amount > balances[_recipient]);
Update Balances
End
Start
Balance Of: view
End
_initialSupply
Pool 4

Figure 4.32: BPMN Diagram for the Basic Token Business Contract

```
pragma solidity >=0.4.0 <0.7.0;

contract BasicToken {
    uint public initialSupply;

    mapping(address=>uint) balances;

    constructor(uint _initialSupply) public {
        initialSupply = _initialSupply;
        balances[msg.sender] = _initialSupply;
    }

    function transfer(address _recipient, uint _amount) public {
        require(balances[msg.sender] >= _amount, "Not enough funds");
        require(_recipient != msg.sender, "No need to send tokens to yourself");
        require(balances[_recipient] + _amount > balances[_recipient]);
        balances[msg.sender] -= _amount;
        balances[_recipient] += _amount;
    }

    function balanceOf(address _owner) public view returns (uint) {
        return balances[_owner];
    }
}
```

Figure 4.33: Re-generated Solidity Code for the Basic Token Business Contract

4.3.3. Crowd Sale

Crowd sale is a process of collecting small amounts of capital from a large number of individuals to finance a new business venture or project. This contract elaborates how the funds been transferred among the beneficiary, backer and contract. This is the most complex diagram (Figure 4.34) which is drawn using the proposed system up to now.

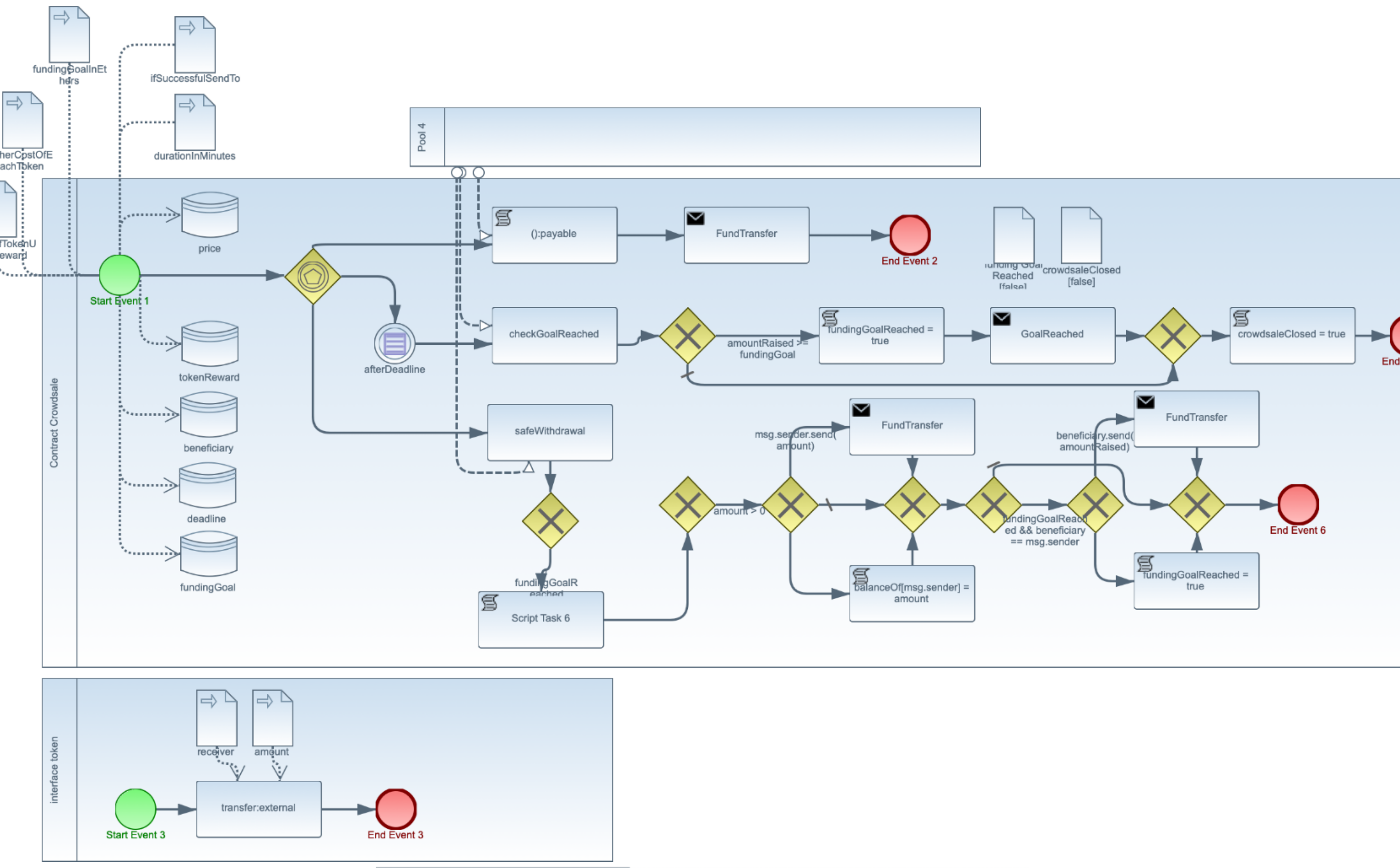


Figure 4.34: BPMN Diagram for the Crowd Sale Business Contract

```
pragma solidity >=0.4.0 <0.7.0;

contract Crowdsale {

    address public beneficiary;
    uint public deadline;
    token public tokenReward;
    uint public price;
    uint public fundingGoal;
    bool crowdsaleClosed = false;
    bool fundingGoalReached = false;

    event FundTransfer(address beneficiary, uint amountRaised, bool fundingGoalReached);
    event GoalReached(uint beneficiary, uint amountRaised);

    constructor (address ifSuccessfulSendTo, uint fundingGoalInEthers, address
addressOfTokenUsedAsReward, uint durationInMinutes, uint etherCostOfEachToken) {
        beneficiary = ifSuccessfulSendTo;
        fundingGoal = fundingGoalInEthers * 1 ether;
        deadline = now + durationInMinutes * 1 minutes;
        price = etherCostOfEachToken * 1 ether;
        tokenReward = token(addressOfTokenUsedAsReward);
    }

    modifier afterDeadline() {
        if (now >= deadline) _;

    }

    function() public payable {
        require(!crowdsaleClosed);
        uint amount = msg.value;
        balanceOf[msg.sender] += amount;
        amountRaised += amount;
        tokenReward.transfer(msg.sender, amount / price);
        emit FundTransfer(beneficiary, amountRaised, false);
    }

    function safeWithdrawal() public {
        if (!fundingGoalReached) {
            uint amount = balanceOf[msg.sender];
            balanceOf[msg.sender] = 0;
            if (amount > 0) {
                if (msg.sender.send(amount)) {
                    emit FundTransfer(beneficiary, amountRaised, false);
                } else {
                    balanceOf[msg.sender] = amount;
                }
            }
        }

        if (fundingGoalReached && beneficiary == msg.sender) {
            if (beneficiary.send(amountRaised)) {
                emit FundTransfer(beneficiary, amountRaised, false);
            } else {
                fundingGoalReached = true;
            }
        }
    }

    function checkGoalReached() public afterDeadline {
        if (amountRaised >= fundingGoal) {
            fundingGoalReached = true
            }

            emit GoalReached(beneficiary, amountRaised);

            crowdsaleClosed = true;
        }

    }

interface token {
    function transfer(address receiver, uint amount) external;
}
```

Figure 4.35: Re-generated Solidity Code for the Crowd Sale Business Contract

4.4. Gap Analysis of the Supported BPMN 2.0 Constructs

A detailed analysis on BPMN 2.0 specification is done to identify the available BPMN 2.0 elements.

Table 4.12: Gap Analysis of the Supported BPMN 2.0 Constructs

BPMN Construct	Description	Support Available?	Remarks
Event	Start Event	Yes	
	Intermediate Event	Yes	
	End Event	Yes	
Event Type	Message, Timer, Error, Escalation, Cancel, Compensation, Conditional, Link, Signal, Terminate, Multiple	Yes	Only supported Conditional, Timer
Activity	Task, Sub-Process, Call Activity, Global Task	Yes	All Activities should be available in the same diagram
Task	User, Manual, Business Rule, Script, Service, Send, Receive	Yes	
Gateway	Exclusive, Inclusive, Parallel, Complex, Event Based	Yes	
Sequence Flow	Normal, Uncontrolled, Conditional, Default, Exception, Compensation	Yes	Except Compensation
Message Flow		Yes	
Association		Yes	
Pool		Yes	
Lane		No	Multiple lanes not supported
Data Object	Data Object, Data Input, Data Output	Yes	
Data Store		Yes	
Message		Yes	
Fork		Yes	

Join		Yes	
Branching Point	Exclusive, Event-Based, Inclusive	Yes	
Merging		Yes	
Looping	Activity, Sub-Process, Sequence Flow	No	
Multiple Instances		No	
Process Break		Yes	Only with conditional event catchers
Transaction		No	
Group		No	
Text Annotation		Yes	Mapped as comments

4.5. Gap Analysis of the Supported Solidity Constructs

The Table 4.13 depicts which Solidity constructs are supported by the Translator when converting from the BPMN 2.0.

Table 4.13: Gap Analysis of the Solidity Language Constructs Supported

Solidity Construct	Description	Supported?	Remarks
Pragma		Yes	Partially supported. Supported static version pragma will be added by the Translator.
Imports		No	Not available through BPMN constructs.
Contract Definitions	State Variables	Yes	Available through Data Objects and Data Stores
	Functions	Yes	Available through Tasks
	Function Modifiers	Yes	Available through Conditional Event Definitions
	Events	Yes	Available through SendTask
	Struct Types	Yes	Available through Data Stores and Data Input Objects
	Enum Types	Yes	Available through DataType metadata in Data Items.
Value Types	Boolean, Integer, Fixed Point Numbers, Address, Arrays, Rational,	Yes	Available through DataType metadata in Data Items.

	String, Hexadecimal		
Function Types		No	
Reference Types	Arrays	Yes	Available through isCollection and DataType metadata in Data Items
	Structs	Yes	Available through Data Stores and Data Input Objects
Mapping Types		Yes	Available through DataType metadata in Data Items
Visibility and Getters	External, public, internal, private	Yes	Derived from the name of the Task; '<TaskName>[:visiblityMod]'
Libraries, Interfaces		Yes	Derived from the Pool Name. 'Interface <PoolName>' or 'Library <PoolName>'
Error Handling		Yes	Partially supported. 'require' is supported.
Control Structures		Yes	Partially supported. Looping constructs yet to support.

4.6. BPMN 2.0 Modelling Recommendations

Granularity of the diagrams should be taken into the account when modelling the diagram. It is important to notice that the BPMN 'Script Task' can be used to add a Task written in a given language (e.g. Java). For instance, BPMN diagram drew in Figure 4.34 in page 101, can be re-drawn as below (Figure 4.36);

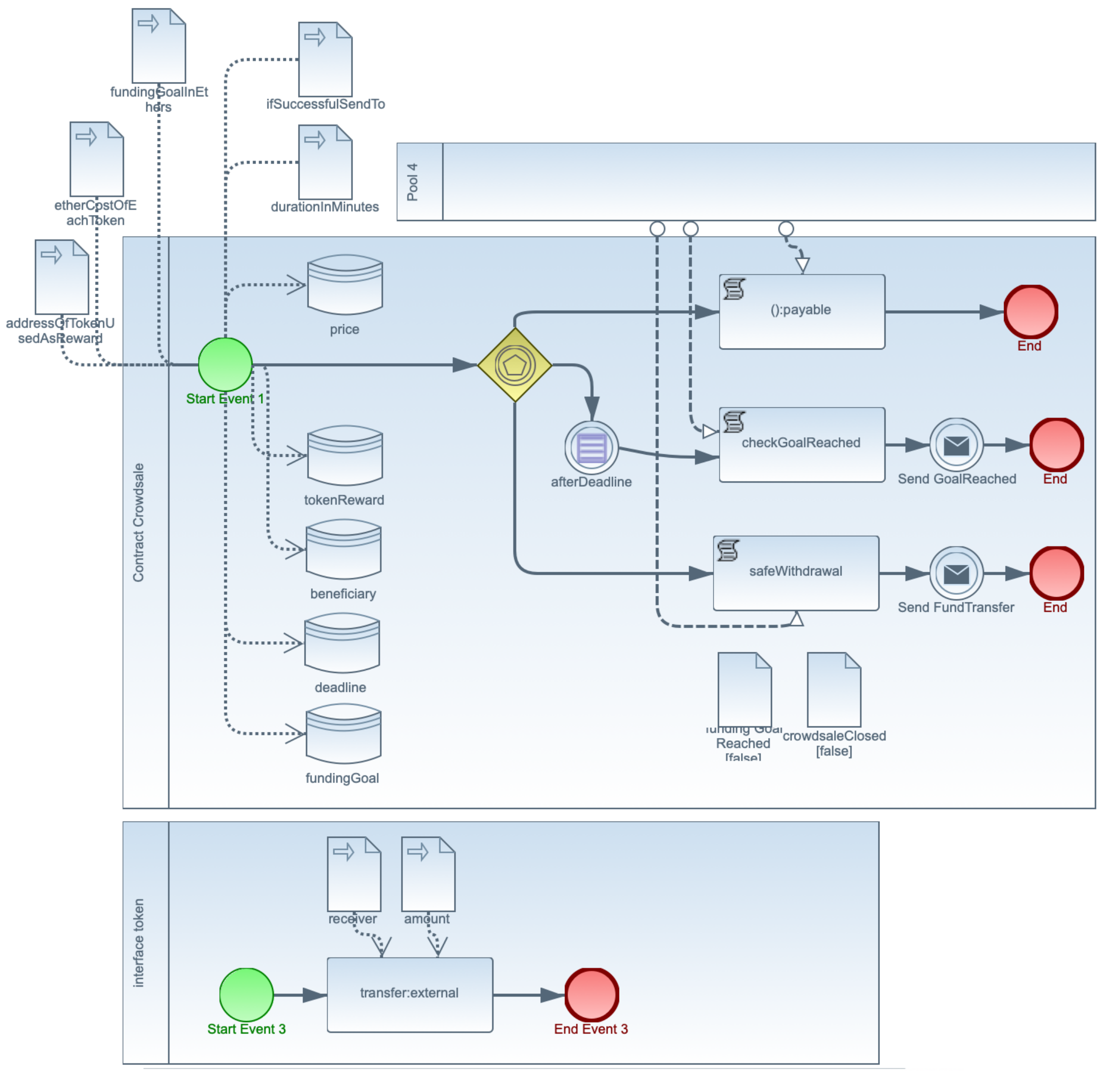


Figure 4.12: Abstract BPMN Diagram for the Crowd Sale

In other words, this is an abstract diagram of the Figure 4.34. Translation of the Figure 4.36, still generates the same Solidity code. However, the readability of the diagram is bit less due to fact that ScriptTasks have replaced all the Gateways that was available in Figure 4.34.

While the ScriptTasks provides an easy way to hide the complexity of the diagram, we recommend to un-hide the decision paths (Exclusive Gateways) from the diagram.

4.4. Chapter Conclusion

This chapter describes the testing and evaluations which were conducted in this research. Results of the developed BPMN-to-Solidity translator and sample BPMN diagrams and the re-generated Solidity codes are included in this chapter. In addition, a detail discussion on gap analysis of the supported BPMN2.0 constructs and Solidity constructs are included.

5. Conclusion and Future Work

5.1. Conclusion

This research addressed the problem of **lack of an efficient translator to reliably convert business process models defined in BPMN into Solidity smart contracts**. Most significant contributions of this research are the mapping rules of the BPMN 2.0 to Solidity language, A Proof-of-Concept (PoC) implementation to prove the derived work and the extensive literature survey on the state-of-the-art related work. The implementation details are freely available in GitHub[11] under the Apache2 opensource-friendly license.

In the literature review chapter, Smart contracts and its applications are discussed. Then Business Process Modelling Notation (BPMN) modelling technique was selected from the various Business Process Modelling (BPM) techniques among Flow Charts, Petri Nets, Data Flow Diagrams (DFDs), Role Activity Diagrams (RADs) as it serves the purpose of this research. 'Eclipse BPMN2.0 Modeller' has been used as the business process modelling tool as part of analysis of the Tools & Plugins for BPMN. Further, detailed analysis on the BPMN2.0 specification and the Solidity language documentation is done.

The main goal of this research is "**to develop an efficient drag-and-drop icon-based translator to convert business process models defined in BPMN into Solidity smart contracts**". The Research Methodology chapter discusses the research approach and steps taken. For the data-collection, primarily, smart contract implementations written in Solidity language were collected and derived the business use cases using BPMN diagrams.

The development of the research consists two main components, (1) BPMN Modeler and (2) BPMN-to-Solidity Translator. For the BPMN Modeler, 'BPMN2 Modeler' Plugin for Eclipse IDE is integrated. For the BPMN-to-Solidity Translator, an ANTLR based compiler is written from the scratch. Input for the compiler is the XML representation of the BPMN diagram and the output is a generated Solidity language contract. Steps followed for the implementation are described in-detail in the Development of the Translator section in page 32. Abstract Syntax Tree (AST) representations for the BPMN and Solidity is introduced to translate a BPMN AST into a Solidity AST, and then finally generate the Solidity code. The diagram representations of the ASTs can be found in Figure 3.12 and Figure 3.13 in page 39.

A particular business transaction can be converted into a smart contract, if and only if that contract can be mapped using the notations in BPMN 2.0 specification and the diagram should be a valid BPMN diagram as the specification. In other words, any business transaction drawn as per the BPMN-to-Solidity mapping specification using a BPMN2 modelling tool can be converted into Solidity code. Incomplete diagrams or diagrams with the incorrect syntaxes will get appropriate error messages.

Further the results of the qualitative survey study suggest that the BPMN-to-Solidity translation is at the satisfactory level. A detailed analysis on this feedback of the study can be found in page 62. Further, the evaluation results of the process model suggest that PoC handles erroneous cases as expected and sufficiently flexible to draw complex diagrams such as 'CrowdSale' example. Moreover, evaluation of the translation verified with the pre-identified abstract features, 'Entry Points', 'Branch Points' and 'Persistent Points'.

Correctness and efficiency are validated with the Truffle tests running on a blockchain test network (refer page 68). In addition, currently available systems are compared against the proposed approach. A Comparative analysis between the Caterpillar and the Proposed approach (in page 72) shows that the proposed approach is generating solidity code that has the minimum lines of code (58 lines) and minimum gas consumption (1,252,037) with a satisfactory code generation time(~263ms). Thus, results of the experiments suggest that the proposed approach is more efficient in converting BPMN to Solidity.

Thus, the goal of this research, *to develop an efficient drag-and-drop icon-based translator to convert business process models defined in BPMN into Solidity smart contracts* is achieved.

5.1. Limitation and Future Work

Current implementation of the BPMN-to-Solidity translator is still in the PoC (Proof-Of-Concept) level and does not address the user-experience aspect in a great extent. Certain steps remain disconnected while translating the BPMN XML file into Solidity code. As the future work of this research, it is expected to develop a tool with attractive user-friendly interfaces to convert business contracts which are drawn using BPMN2.0 to Solidity smart contracts addressing aforementioned issues.

For the evaluation of the Solidity codes generated by the XML representation of the BPMN diagrams, six use cases of different complexities from different domains has been selected. Thus, the generalizability of the BPMN-to-Solidity translator introduced in this research is limited to the scope of the use cases provided in page **Error! Bookmark not defined.**. BPMN elements coverage for contracts are listed in **Error! Reference source not found.** in page **Error! Bookmark not defined.**. However, in theory; once the BPMN-to-Solidity translator supports all elements listed in BPMN2.0 specification; it should be able to produce a valid Solidity smart contract code for any given BPMN XML.

Moreover, the current study only focused on basic control-flow patterns and not all the elements in BPMN 2.0 specification are covered. Thus, need to cover the full BPMN2.0 specification to proclaim the full BMPN2.0 support. The gap analysis done in section 4.4 (page 92) clearly identifies the pending BPMN elements. Nevertheless, the available BPMN elements sufficiently covers the basic control-flow patterns of the process modelling [17].

Writing tests to verify the business logic also requires a significant effort. Thus, as a future work, the translator should be able to generate the test templates for the generated contracts minimizing this tedious effort. Integrating a test-framework such as Truffle will provide a runtime for the contracts and allow the translator to deploy the contracts & tests in the test networks as well.

The participation count of the qualitative survey is very limited (N=7). In the future, participant count can be increased to get a rich feedback. However, one advantage of this limited participation is the chance of getting feedback considering the quality over quantity.

In addition, the feedback received through the qualitative survey study (refer Appendix E, page 150) signifies that there are several improvements needs to be done. There's a room for the improvement in the representation for the complex diagrams. Also, the state management of the generated contract is need to be improved without depending on the process designer to implement the additional checks. Moreover, as per the feedback traceability and readability of the generated code can be further improved by adding compiler generated comments.

A limited support available for the reusability of the same code-blocks across the multiple contracts. This is allowed through sharing the Tasks across multiple Pools in BPMN terms. However, still the translator supports single file generation as an output. Thus, sharable code-blocks are limited to the scope of a single file.

Furthermore, when comparing with the existing systems such as Caterpillar, the comparisons are not straight forward since the purpose and the audience of these tools and systems are different. For instance, Caterpillar is a BPMN execution engine and provides state monitoring and other value-added features in addition. Compared to the proposed approach, it provides the raw Solidity code contract that needs to be deployed in any blockchain of the users need. Nevertheless, since Caterpillar is also an opensource BPMN execution engine, In the future, we might be able to integrate the proposed BPMN-to-Soldity translator as a core-component to Caterpillar execution engine to enhance their processes.

References

[1] A. J. Bellia, “Promises, Trust, and Contract Law,” *47 AM.J.JURIS.,* vol. 25, pp. 25-26, 2002.

[2] G. Valentina, L. Fabrizio, D. Claudio, P. Chiara and S. Víctor, “Blockchain and Smart Contracts for Insurance: Is the Technology Mature Enough?,” MDPI, Basel, Switzerland, 2018.

[3] ChainTrade, “10 Advantages of Using Smart Contracts,” Medium, 27 Dec 2017. [Online]. Available: https://medium.com/@ChainTrade/10-advantages-of-using-smart-contracts-bc29c508691a. [Accessed 25 June 2018].

[4] “Solidity,” 2018. [Online]. Available: https://solidity.readthedocs.io/en/v0.4.25/. [Accessed 26 11 2018].

[5] A. C. Paulus, “Implementation of Blockchain Powered Smart Contracts in Governmental Services,” Delft University of Technology, 2018.

[6] Lucid Software Inc., “What is Business Process Modeling Notation,” Lucidchart, 2019. [Online]. Available: https://www.lucidchart.com/pages/bpmn. [Accessed 1 June 2019].

[7] J. Ladleif, M. Weske and I. Weber, “Modeling and Enforcing Blockchain-Based Choreographies,” *BPM 2019,* vol. 11675, no. Lecture Notes in Computer Science, pp. 69-85, 2019.

[8] L. Severeijns, “What is blockchain? How is it going to affect Business?,” Vrije Universiteit , Amsterdam, 2017.

[9] ISCA, “Blockchain: Re-imagining Multi-Party Transactions for Businesses,” Institute of Singapore Chartered Accountants, Singapore , 2017 .

[10] L. Aldin and S. d. Cesare, “A Comparative Analysis Of Business Process Modelling Techniques,” in *U.K. Academy for Information Systems (UKAIS 2009), 14th Annual Conference*, UK, 2009.

[11] R. Koncevičs, L. Peņicina, A. Gaidukovs, M. Darģis, R. Burbo and A. Auziņš, “Comparative Analysis of Business Process Modelling Tools for Compliance Management Support,” *The Journal of Riga Technical University,* vol. 21, pp. 22-27, 2017.

[12] T. K. Sharma, “WHAT IS SOLIDITY, PROGRAMMING LANGUAGE FOR ETHEREUM SMART CONTRACTS?,” Blockchain Council, 2 September 2017. [Online]. Available: https://www.blockchain-council.org/ethereum/what-is-solidity-programming-language-for-ethereum-smart-contracts/. [Accessed 2 June 2019].

[13] O. López-Pintado, B. García-Bañuelos, M. Dumas and I. Weber, “Caterpillar: A Blockchain-Based BusinessProcess Management System,” in *Proceedings of the BPM Demo Track and BPM Dissertation Award co-locatedwith 15th International Conference on Business Process Modeling (BPM 2017),*, Barcelona, Spain, eptember 13, 2017..

[14] O. López-Pintado, L. García-Bañuelos, M. Dumas and I. Weber, “CATERPILLAR: A Business Process Execution Engine on the Ethereum Block,” *Software: Practice and Experience,* no. 00, pp. 01-45, 2018.

[15] S. Schmidt and M. Jung, “The unified framework for blockchain based business integration,” Unibright, 2018.

[16] L. García-Bañuelos, A. Ponomarev, M. Dumas and I. Weber, “Optimized Execution of Business Processes on Blockchain,” in *Business Process Management: 15th International Conference*, Barcelona, Spain, 2017.

[17] I. Weber, X. Xu, R. Riveret, G. Governatori, A. Ponomarev and J. Mendling, “Untrusted Business Process Monitoring and Execution Using Blockchain,” in *BPM 2016*, Rio de Janeiro, Brazil, Springer, Cham, Sept. 2016, pp. 329-347.

[18] I. Weber, X. Xu, R. Riveret, G. Governatori, A. Ponomarev and J. Mendling, “Using Blockchain to Enable Untrusted Business Process Monitoring and Execution, Technical Report UNSW-CSE-TR-201609,” University of New South Wales, 2016.

[19] “The usage of BPMN library to define workflow,” 03 01 2017. [Online]. Available: https://dspace.cvut.cz/bitstream/handle/10467/66832/F3-BP-2017-Brichkova-Evgeniya-The_usage_of_BPMN_library_to_define_workflow.pdf. [Accessed 20 04 2019].

[20] M. Muehlen zur and J. Recker, “How Much Language is Enough? Theoretical and Practical Use of the Business Process Modeling Notation.,” in *In Proc. CAiSE*, 2008.

[21] P. Hegedűs, “Towards Analyzing the Complexity Landscape of Solidity Based Ethereum Smart Contracts,” *MTA-SZTE Research Group on Artificial Intelligence,* vol. 7, no. 1, p. 6, 2019.

[22] L. M. Laird and M. C. Brennan, “ Cyclomatic Complexity,” in *Software Measurement and Estimation: A Practical Approach*, New Jersey, A John Wiley & Sons, Inc., 2006, pp. 58-62.

[23] S. J. Naqvi, “Converting a Property Rental Paper Contract into a Smart Contract,” Medium, 24 April 2017. [Online]. Available: https://medium.com/@naqvi.jafar91/converting-a-property-rental-paper-contract-into-a-smart-contract-daa054fdf8a7. [Accessed 1 June 2019].

[24] Q. Fang, “shares-contract,” Github, 23 April 2018. [Online]. Available: https://github.com/qimingfang/shares-contract. [Accessed 1 May 2019].

[25] P. Brudny, “learning-solidity-2018,” Medium, 1 August 2018. [Online]. Available: https://github.com/pbrudny/learning-solidity-2018. [Accessed 4 April 2019].

[26] “Multi Party Settlement,” Merit Systems Private Limited, [Online]. Available: http://meritsystems.com/multi-party-settlement/. [Accessed 26 June 2018].

[27] A. Awaysheh and R. D. Klassen, “The impact of supply chain structure on the use of supplier socially responsible practices,” *International Journal of Operations & Production Management,* vol. 30, no. 12, pp. 1246-1268, 2010.

[28] K. Francisco and D. Swanson, “The Supply Chain Has No Clothes: Technology Adoption of Blockchain for Supply Chain Transparency,”

Department of Marketing & Logistics, University of North Florida, Jacksonville, 2018.

[29] A. Wright and P. De Filippi, “Decentralized Blockchain Technology and the Rise of Lex Cryptographia,” p. 58, 10 March 2015.

[30] S. Seebacher and R. Schuritz, “Blockchain Technology as an Enabler of Service Systems: A Structured Literature Review,” in *The International Conference on Exploring Services Science*, Rome, 2017.

[31] G. V. Research, “Blockchain Technology Market Size, Share & Trends Analysis Report By Type (Public, Private, Hybrid), By Application (Financial Services, Consumer Products, Technology, Telecom), And Segment Forecasts, 2018 - 2024,” San Francisco, United States, 2018.

[32] V. Buterin, “Ethereum White Paper : A NEXT GENERATION SMART CONTRACT & DECENTRALIZED APPLICATION PLATFORM,” 2014.

[33] Object Management Group (OMG), “Business Process Model and Notation (BPMN),” OMG, 2013.

5. Appendices

Appendix A: Unibright Workflow Designer Generated Solidity Code

A1. Unibright Workflow Designer generated Solidity Code for the Demo Workflow Example

```
pragma solidity ^0.4.24;

// Represents the base of a unibright contract
contract UnibrightContract {

    // the owner of the contract
    address public owner;

    // state of the contract
    ContractState public contractState;

    // states the contract can be in
    enum ContractState { CREATED, PUBLISHED, RUNNING, CANCELLED, FINISHED }

    // A modifier that can limit the execution of a function to a specific
```

```
address
    modifier onlyBy(address _account) {
        require(msg.sender == _account, "Not allowed");
        _;
    }

    // A modifier that can limit the execution of a function to a specific
contract state
    modifier onlyInState(ContractState state) {
        require(contractState == state, string(abi.encodePacked("Only allowed
when contract is in state ", state)));
        _;
    }

    constructor() public {
        owner = msg.sender;
    }

    // publish is called by the owner of the contract to progress to the state
PUBLISHED
    function publish() public onlyBy(owner) {
        contractState = ContractState.PUBLISHED;
    }

    // start is called by the owner of the contract to progress to the state
RUNNING
    function start() public onlyBy(owner) {
        contractState = ContractState.RUNNING;
    }

    // cancel is called by the owner of the contract to progress to the state
cancelled
    function cancel() public onlyBy(owner) {
        contractState = ContractState.CANCELLED;
    }

    // finish is called to progress to the state FINISHED
    function finish() private {
        contractState = ContractState.FINISHED;
    }
}
pragma solidity ^0.4.24;

import "./UnibrightContract.sol";

contract MultiPartyApproval is UnibrightContract {

    // the status of the whole approval process
    Status public approvalStatus;

    // holds the timestamp of the last update
    uint public lastUpdate;

    // states the contract and the feedbacks of the approvers can be in
    enum Status { UNDEFINED, NEEDSWORK, APPROVED }

    // uuid of the end node
    bytes16 private endId;

    // map of uuids to approvers, filled during the deployment
    mapping(bytes16 => Approver) private approvers;

    // map of uuids to feedbacks, filled during the deployment
    mapping(bytes16 => Feedback) private feedbacks;

    // map of addresses to uuids, filled during the setup phase
    mapping(address => bytes16) private addresses;

    // Approver holds information about an Approver
    struct Approver {
        Status status;          // the status of the approval
        string comment;         // a description related to the approval status
        uint approvalDate;      // time at which the approver gave his approval

        uint feedbacksNeeded;   // remaining number of feedbacks needed for this
approver to give his approval
        bytes16 feedbackId;     // the id of the feedback object attached to this
approver

        bool exists;            // required to check if mapping exists
    }

    // Feedback holds information about the feedback given by an group of
approvers
    struct Feedback {
        Status status;          // the status of the feedback node

        uint approvalsGiven;    // number of approvals given for this feedback
node
        uint approvalsNeeded;   // number of approvals needed for this feedback
node to set its status
        bytes16[] approverIds;  // ids of successor approvers attached to the
feedback

        bool exists;            // required to check if mapping exists
    }

    constructor() public {
        approvalStatus = Status.UNDEFINED;
        publish();

        start();
    }

    // Adds the Approver node to the MPA network that represents the end of the
process
    function addEndNode(bytes16 endNodeId, uint feedbacksNeeded) public
onlyBy(owner) onlyInState(ContractState.PUBLISHED) {
        addApprover(endNodeId, 0, feedbacksNeeded);
        endId = endNodeId;
    }

    // Add an approver to the MPA network
    function addApprover(bytes16 approverId, bytes16 feedbackId, uint
feedbacksNeeded) public
    onlyBy(owner) onlyInState(ContractState.PUBLISHED) {

        require(!approvers[approverId].exists, "approver already exists");

        Approver memory approver = Approver({
            status: Status.UNDEFINED,
            comment: "",
            approvalDate: 0,
            feedbackId: feedbackId,
            feedbacksNeeded: feedbacksNeeded,
            exists: true
            });
        approvers[approverId] = approver;
    }

    // Set the address for an approver
    function setApproverAddress(address approverAddress, bytes16 approverId)
```

```
public onlyBy(owner) onlyInState(ContractState.PUBLISHED) {
        require(approvers[approverId].exists, "no approver with given id found");
        addresses[approverAddress] = approverId;
    }

    // returns the approver for a given uuid
    function getApprover(bytes16 id) public view returns (Status, string, uint,
uint, bytes16) {
        Approver memory approver = approvers[id];
        require(approver.exists, "this approver does not exist.");
        return (
        approver.status,
        approver.comment,
        approver.approvalDate,
        approver.feedbacksNeeded,
        approver.feedbackId
        );
    }

    // Add a feedback node to the MPA network
    function addFeedback(bytes16 feedbackId, uint approvalsNeeded, bytes16[]
approverIds) public
    onlyBy(owner) onlyInState(ContractState.PUBLISHED) {

        require(!feedbacks[feedbackId].exists, "a feedback object with this id
already exists");

        Feedback memory feedback = Feedback({
            status: Status.UNDEFINED,
            approvalsGiven: 0,
            approvalsNeeded: approvalsNeeded,
            approverIds: approverIds,
            exists: true
            });
        feedbacks[feedbackId] = feedback;
    }

    // returns the feedback note for the given uuid
    function getFeedback(bytes16 id) public view returns (Status, uint, uint,
bytes16[]) {
        require(feedbacks[id].exists, "this feedbackId does not exist.");
        return (feedbacks[id].status, feedbacks[id].approvalsGiven,
feedbacks[id].approvalsNeeded, feedbacks[id].approverIds);
    }

    // the actual method called by the approvers to give their feedback
    function giveFeedback(Status status, string comment) public
onlyInState(ContractState.RUNNING) {

        Approver storage approver = approvers[addresses[msg.sender]];
        require(approver.exists, "not allowed to provide feedback");
        require(approver.feedbacksNeeded == 0, "not allowed to provide approval
yet");

        require(status != Status.UNDEFINED, "approval status cannot be set to
undefined");
        require(approver.status == Status.UNDEFINED, "approval already
submitted");

        Feedback storage feedback = feedbacks[approver.feedbackId];
        require(feedback.status != Status.NEEDSWORK, "feedback already set");
        require(feedback.approvalsGiven < feedback.approvalsNeeded, "feedback
already set");

        approver.status = status;
        approver.comment = comment;
        approver.approvalDate = now;
        lastUpdate = now;

        // if any approver gives the state NEEDSWORK the contract is finished
        if (status == Status.NEEDSWORK) {
            feedback.status = status;
            approvalStatus = status;
            finish();
            return;
        }

        feedback.approvalsGiven++;

        // if enough approvals have been given for this feedback, reduce the
feedbacksNeeded counter further down the MPANet
        if (feedback.approvalsGiven == feedback.approvalsNeeded) {
            feedback.status = status;

            for (uint i = 0; i < feedback.approverIds.length; i++) {
                bytes16 id = feedback.approverIds[i];
                approvers[id].feedbacksNeeded--;
            }
        }

        // if all feedbacks attached to the end node are set, finish the contract
        if (approvers[endId].feedbacksNeeded == 0) {
            approvalStatus = status;
            finish();
        }
    }

    function publish() public onlyBy(owner) {
        super.publish();
        lastUpdate = now;


        addApprover(0xf621af583a18c245b20571c400222fbf,
0x13587eef37d21840a982743534ffdf9a, 0);
        addApprover(0x5484342aa23f9a46aa631e626b3a3480,
0x0ebf120626a889409228a3b88128020e, 1);
        addEndNode(0x98cb347699cccf42be44ad42b55b91d5, 1);
        addApprover(0xd30cbdf0e7ca5f47acb9168910bd8c09,
0x13587eef37d21840a982743534ffdf9a, 0);
        addApprover(0x5b0fe707b73889448efc9c514c9dd881,
0x13587eef37d21840a982743534ffdf9a, 0);

        bytes16[] memory approverList1 = new bytes16[](1);
        approverList1[0] = 0x5484342aa23f9a46aa631e626b3a3480;
        addFeedback(0x13587eef37d21840a982743534ffdf9a, 1, approverList1);
        bytes16[] memory approverList3 = new bytes16[](1);
        approverList3[0] = 0x98cb347699cccf42be44ad42b55b91d5;
        addFeedback(0x0ebf120626a889409228a3b88128020e, 1, approverList3);


    }

    function start() public onlyBy(owner) onlyInState(ContractState.PUBLISHED) {
        super.start();
        lastUpdate = now;

        //%%CODEGENERATED_IN_PROGRESS%%
    }

    function cancel() public onlyBy(owner) onlyInState(ContractState.RUNNING) {
        super.cancel();
        lastUpdate = now;
        //%%CODEGENERATED_ABORT%%
    }

    function finish() private onlyInState(ContractState.RUNNING) {
        contractState = ContractState.FINISHED;
        lastUpdate = now;

        //%%CODEGENERATED_FINISH%%
    }
}
```

Appendix B: Process Model XML Source for the Loan Assessment Example using Caterpillar and Proposed Approach

B1. Process Model XML Source for the Loan Assessment Example Using Caterpillar v3.0

Example from Caterpillar v1.0 GitHub[11] has been slightly modified to work with Caterpillar v3.0. Changes are done for the document attribute of the BPMN elements using Solidity annotations specification.

```
<?xml version="1.0" encoding="UTF-8"?>
<bpmn:definitions xmlns:bpmn="http://www.omg.org/spec/BPMN/20100524/MODEL" xmlns:
bpmndi="http://www.omg.org/spec/BPMN/20100524/DI" xmlns:di="http://www.omg.org/sp
ec/DD/20100524/DI" xmlns:dc="http://www.omg.org/spec/DD/20100524/DC" xmlns:camund
a="http://camunda.org/schema/1.0/bpmn" xmlns:xsi="http://www.w3.org/2001/XMLSchem
a-
instance" id="Definitions_1" targetNamespace="http://bpmn.io/schema/bpmn" exporte
r="Camunda Modeler" exporterVersion="1.6.0">
  <bpmn:process id="BPM17_Running_Example" name="BPM17_Running_Example" isExecuta
ble="false">
    <bpmn:documentation><![CDATA[bool applicantEligible;
uint monthlyRevenue;
uint loanAmount;
uint cost;
uint appraisePropertyResult;
uint assessLoanRiskResult;
]]></bpmn:documentation>
    <bpmn:startEvent id="StartEvent_1">
      <bpmn:outgoing>SequenceFlow_16ew9vc</bpmn:outgoing>
    </bpmn:startEvent>
    <bpmn:sequenceFlow id="SequenceFlow_16ew9vc" sourceRef="StartEvent_1" targetR
ef="Task_06xlgcp" />
    <bpmn:userTask id="Task_06xlgcp" name="Enter Loan Application">
      <bpmn:documentation><![CDATA[@ Manager @
() : (uint _monthlyRevenue, uint _loanAmount, uint _cost) -
> monthlyRevenue = _monthlyRevenue; loanAmount = _loanAmount; cost = _cost;]]></b
pmn:documentation>
      <bpmn:incoming>SequenceFlow_16ew9vc</bpmn:incoming>
      <bpmn:outgoing>SequenceFlow_1jygpfu</bpmn:outgoing>
    </bpmn:userTask>
    <bpmn:serviceTask id="Task_1aqv42f" name="Assess Loan Risk" camunda:expressio
n="uint monthlyRevenue, uint loadAmount">
      <bpmn:documentation><![CDATA[@ Oracle @
() : (uint _assessLoanRiskResult) -
> assessLoanRiskResult = _assessLoanRiskResult;]]></bpmn:documentation>
      <bpmn:extensionElements>
        <camunda:inputOutput>
          <camunda:inputParameter name="Input_2k32da1">{name:Assess_Loan_Risk}</c
amunda:inputParameter>
          <camunda:outputParameter name="Output_284ft88" />
        </camunda:inputOutput>
      </bpmn:extensionElements>
      <bpmn:incoming>SequenceFlow_0b6dfgq</bpmn:incoming>
      <bpmn:outgoing>SequenceFlow_1splpa9</bpmn:outgoing>
    </bpmn:serviceTask>
    <bpmn:sequenceFlow id="SequenceFlow_1splpa9" sourceRef="Task_1aqv42f" targetR
ef="ExclusiveGateway_18clflo" />
    <bpmn:scriptTask id="Task_1ggq6sf" name="Assess Elegibility">
      <bpmn:incoming>SequenceFlow_1kpqxh8</bpmn:incoming>
      <bpmn:outgoing>SequenceFlow_0ensspb</bpmn:outgoing>
      <bpmn:script><![CDATA[if (assessLoanRiskResult >= appraisePropertyResult)
  applicantEligible = true;
else
  applicantEligible = false;]]></bpmn:script>
    </bpmn:scriptTask>
    <bpmn:exclusiveGateway id="ExclusiveGateway_06dboho" default="SequenceFlow_0v
bi21y">
      <bpmn:incoming>SequenceFlow_0ensspb</bpmn:incoming>
      <bpmn:outgoing>SequenceFlow_069rxq2</bpmn:outgoing>
      <bpmn:outgoing>SequenceFlow_0vbi21y</bpmn:outgoing>
    </bpmn:exclusiveGateway>
    <bpmn:sequenceFlow id="SequenceFlow_0ensspb" sourceRef="Task_1ggq6sf" targetR
ef="ExclusiveGateway_06dboho" />
    <bpmn:sequenceFlow id="SequenceFlow_069rxq2" sourceRef="ExclusiveGateway_06db
oho" targetRef="IntermediateThrowEvent_0k5vl3c">
      <bpmn:conditionExpression xsi:type="bpmn:tFormalExpression">applicantEligib
le</bpmn:conditionExpression>
    </bpmn:sequenceFlow>
    <bpmn:endEvent id="EndEvent_19xiayo" name="Loan app accepted">
      <bpmn:incoming>SequenceFlow_0jigqn5</bpmn:incoming>
      <bpmn:messageEventDefinition />
    </bpmn:endEvent>
    <bpmn:sequenceFlow id="SequenceFlow_0vbi21y" sourceRef="ExclusiveGateway_06db
oho" targetRef="EndEvent_1ubxmre" />
    <bpmn:endEvent id="EndEvent_1ubxmre" name="Loan app rejected">
```

```
      <bpmn:incoming>SequenceFlow_0vbi21y</bpmn:incoming>
      <bpmn:incoming>SequenceFlow_04nl5rk</bpmn:incoming>
      <bpmn:messageEventDefinition />
    </bpmn:endEvent>
    <bpmn:sequenceFlow id="SequenceFlow_1jygpfu" sourceRef="Task_06xlgcp" targetR
ef="ExclusiveGateway_0o4nv8y" />
    <bpmn:sequenceFlow id="SequenceFlow_0b6dfgq" sourceRef="ExclusiveGateway_0o4n
v8y" targetRef="Task_1aqv42f" />
    <bpmn:sequenceFlow id="SequenceFlow_1kpqxh8" sourceRef="ExclusiveGateway_18cl
flo" targetRef="Task_1ggq6sf" />
    <bpmn:parallelGateway id="ExclusiveGateway_0o4nv8y">
      <bpmn:incoming>SequenceFlow_1jygpfu</bpmn:incoming>
      <bpmn:outgoing>SequenceFlow_0b6dfgq</bpmn:outgoing>
      <bpmn:outgoing>SequenceFlow_1pm1lpg</bpmn:outgoing>
    </bpmn:parallelGateway>
    <bpmn:parallelGateway id="ExclusiveGateway_18clflo">
      <bpmn:incoming>SequenceFlow_1splpa9</bpmn:incoming>
      <bpmn:incoming>SequenceFlow_05u8ux5</bpmn:incoming>
      <bpmn:outgoing>SequenceFlow_1kpqxh8</bpmn:outgoing>
    </bpmn:parallelGateway>
    <bpmn:sequenceFlow id="SequenceFlow_1pm1lpg" sourceRef="ExclusiveGateway_0o4n
v8y" targetRef="Task_1n2g1ih" />
    <bpmn:sequenceFlow id="SequenceFlow_05u8ux5" sourceRef="Task_1n2g1ih" targetR
ef="ExclusiveGateway_18clflo" />
    <bpmn:serviceTask id="Task_1n2g1ih" name="Appraise Property" camunda:expressi
on="a = 3">
      <bpmn:documentation><![CDATA[@ Oracle @
(): (uint _appraisePropertyResult) -
> appraisePropertyResult = _appraisePropertyResult;]]></bpmn:documentation>
      <bpmn:extensionElements>
        <camunda:inputOutput>
          <camunda:inputParameter name="Input_264oln8">{name:Appraise_Property}</
camunda:inputParameter>
        </camunda:inputOutput>
      </bpmn:extensionElements>
      <bpmn:incoming>SequenceFlow_1pm1lpg</bpmn:incoming>
      <bpmn:outgoing>SequenceFlow_05u8ux5</bpmn:outgoing>
    </bpmn:serviceTask>
    <bpmn:sequenceFlow id="SequenceFlow_1pxsdl6" sourceRef="Task_15lfaes" targetR
ef="ExclusiveGateway_0ga7p17" />
    <bpmn:userTask id="Task_15lfaes" name="Confirm Acceptance">
      <bpmn:documentation><![CDATA[@ Manager @
() : (bool _confirmation) -
> applicantEligible = _confirmation; ]]></bpmn:documentation>
      <bpmn:incoming>SequenceFlow_14exagm</bpmn:incoming>
      <bpmn:outgoing>SequenceFlow_1pxsdl6</bpmn:outgoing>
    </bpmn:userTask>
    <bpmn:exclusiveGateway id="ExclusiveGateway_0ga7p17" default="SequenceFlow_04
nl5rk">
      <bpmn:incoming>SequenceFlow_1pxsdl6</bpmn:incoming>
      <bpmn:outgoing>SequenceFlow_0jigqn5</bpmn:outgoing>
      <bpmn:outgoing>SequenceFlow_04nl5rk</bpmn:outgoing>
    </bpmn:exclusiveGateway>
    <bpmn:sequenceFlow id="SequenceFlow_0jigqn5" sourceRef="ExclusiveGateway_0ga7
p17" targetRef="EndEvent_19xiayo">
      <bpmn:conditionExpression xsi:type="bpmn:tFormalExpression">applicantEligib
le</bpmn:conditionExpression>
    </bpmn:sequenceFlow>
    <bpmn:sequenceFlow id="SequenceFlow_04nl5rk" sourceRef="ExclusiveGateway_0ga7
p17" targetRef="EndEvent_1ubxmre" />
    <bpmn:sequenceFlow id="SequenceFlow_14exagm" sourceRef="IntermediateThrowEven
t_0k5vl3c" targetRef="Task_15lfaes" />
    <bpmn:intermediateThrowEvent id="IntermediateThrowEvent_0k5vl3c" name="Confir
mation request sent">
      <bpmn:incoming>SequenceFlow_069rxq2</bpmn:incoming>
      <bpmn:outgoing>SequenceFlow_14exagm</bpmn:outgoing>
      <bpmn:messageEventDefinition />
    </bpmn:intermediateThrowEvent>
  </bpmn:process>
  <bpmndi:BPMNDiagram id="BPMNDiagram_1">
    <bpmndi:BPMNPlane id="BPMNPlane_1" bpmnElement="BPM17_Running_Example">
      <bpmndi:BPMNShape id="_BPMNShape_StartEvent_2" bpmnElement="StartEvent_1">
        <dc:Bounds x="132" y="218" width="36" height="36" />
        <bpmndi:BPMNLabel>
          <dc:Bounds x="105" y="254" width="90" height="0" />
        </bpmndi:BPMNLabel>
      </bpmndi:BPMNShape>
      <bpmndi:BPMNEdge id="SequenceFlow_16ew9vc_di" bpmnElement="SequenceFlow_16e
w9vc">
        <di:waypoint xsi:type="dc:Point" x="168" y="236" />
        <di:waypoint xsi:type="dc:Point" x="196" y="236" />
        <bpmndi:BPMNLabel>
          <dc:Bounds x="137" y="221" width="90" height="0" />
        </bpmndi:BPMNLabel>
      </bpmndi:BPMNEdge>
      <bpmndi:BPMNShape id="UserTask_0g4b87e_di" bpmnElement="Task_06xlgcp">
        <dc:Bounds x="196" y="196" width="100" height="80" />
      </bpmndi:BPMNShape>
      <bpmndi:BPMNShape id="ServiceTask_1rwb825_di" bpmnElement="Task_1aqv42f">
        <dc:Bounds x="383" y="142" width="100" height="80" />
      </bpmndi:BPMNShape>
      <bpmndi:BPMNEdge id="SequenceFlow_1splpa9_di" bpmnElement="SequenceFlow_1sp
lpa9">
        <di:waypoint xsi:type="dc:Point" x="483" y="182" />
        <di:waypoint xsi:type="dc:Point" x="523" y="182" />
        <di:waypoint xsi:type="dc:Point" x="523" y="211" />
        <bpmndi:BPMNLabel>
          <dc:Bounds x="458" y="167" width="90" height="0" />
        </bpmndi:BPMNLabel>
      </bpmndi:BPMNEdge>
      <bpmndi:BPMNShape id="ScriptTask_0jlspho_di" bpmnElement="Task_1ggq6sf">
        <dc:Bounds x="583" y="196" width="100" height="80" />
      </bpmndi:BPMNShape>
      <bpmndi:BPMNShape id="ExclusiveGateway_06dboho_di" bpmnElement="ExclusiveGa
teway_06dboho" isMarkerVisible="true">
        <dc:Bounds x="721" y="211" width="50" height="50" />
        <bpmndi:BPMNLabel>
          <dc:Bounds x="701" y="261" width="90" height="0" />
        </bpmndi:BPMNLabel>
      </bpmndi:BPMNShape>
      <bpmndi:BPMNEdge id="SequenceFlow_0enspb_di" bpmnElement="SequenceFlow_0en
sspb">
        <di:waypoint xsi:type="dc:Point" x="683" y="236" />
        <di:waypoint xsi:type="dc:Point" x="721" y="236" />
        <bpmndi:BPMNLabel>
          <dc:Bounds x="657" y="221" width="90" height="0" />
        </bpmndi:BPMNLabel>
      </bpmndi:BPMNEdge>
      <bpmndi:BPMNEdge id="SequenceFlow_069rxq2_di" bpmnElement="SequenceFlow_069
rxq2">
        <di:waypoint xsi:type="dc:Point" x="746" y="211" />
        <di:waypoint xsi:type="dc:Point" x="746" y="160" />
        <di:waypoint xsi:type="dc:Point" x="774" y="160" />
        <bpmndi:BPMNLabel>
          <dc:Bounds x="716" y="185.5" width="90" height="0" />
        </bpmndi:BPMNLabel>
      </bpmndi:BPMNEdge>
      <bpmndi:BPMNShape id="EndEvent_0f5ggdp_di" bpmnElement="EndEvent_19xiayo">
        <dc:Bounds x="1046" y="142" width="36" height="36" />
        <bpmndi:BPMNLabel>
          <dc:Bounds x="1019" y="178" width="90" height="25" />
        </bpmndi:BPMNLabel>
      </bpmndi:BPMNShape>
      <bpmndi:BPMNEdge id="SequenceFlow_0vbi21y_di" bpmnElement="SequenceFlow_0vb
i21y">
        <di:waypoint xsi:type="dc:Point" x="746" y="261" />
        <di:waypoint xsi:type="dc:Point" x="746" y="302" />
        <di:waypoint xsi:type="dc:Point" x="968" y="302" />
        <bpmndi:BPMNLabel>
          <dc:Bounds x="716" y="281.5" width="90" height="0" />
        </bpmndi:BPMNLabel>
      </bpmndi:BPMNEdge>
      <bpmndi:BPMNShape id="EndEvent_1hyovvg_di" bpmnElement="EndEvent_1ubxmre">
        <dc:Bounds x="968" y="284" width="36" height="36" />
        <bpmndi:BPMNLabel>
          <dc:Bounds x="941" y="320" width="90" height="13" />
        </bpmndi:BPMNLabel>
      </bpmndi:BPMNShape>
      <bpmndi:BPMNEdge id="SequenceFlow_1jygpfu_di" bpmnElement="SequenceFlow_1jy
gpfu">
        <di:waypoint xsi:type="dc:Point" x="296" y="236" />
        <di:waypoint xsi:type="dc:Point" x="321" y="236" />
        <bpmndi:BPMNLabel>
```

```
          <dc:Bounds x="263.5" y="221" width="90" height="0" />
        </bpmndi:BPMNLabel>
      </bpmndi:BPMNEdge>
      <bpmndi:BPMNEdge id="SequenceFlow_0b6dfgq_di" bpmnElement="SequenceFlow_0b6
dfgq">
        <di:waypoint xsi:type="dc:Point" x="346" y="211" />
        <di:waypoint xsi:type="dc:Point" x="346" y="182" />
        <di:waypoint xsi:type="dc:Point" x="383" y="182" />
        <bpmndi:BPMNLabel>
          <dc:Bounds x="316" y="196.5" width="90" height="0" />
        </bpmndi:BPMNLabel>
      </bpmndi:BPMNEdge>
      <bpmndi:BPMNEdge id="SequenceFlow_1kpqxh8_di" bpmnElement="SequenceFlow_1kp
qxh8">
        <di:waypoint xsi:type="dc:Point" x="548" y="236" />
        <di:waypoint xsi:type="dc:Point" x="583" y="236" />
        <bpmndi:BPMNLabel>
          <dc:Bounds x="520.5" y="221" width="90" height="0" />
        </bpmndi:BPMNLabel>
      </bpmndi:BPMNEdge>
      <bpmndi:BPMNShape id="ParallelGateway_1567msb_di" bpmnElement="ExclusiveGat
eway_0o4nv8y">
        <dc:Bounds x="321" y="211" width="50" height="50" />
        <bpmndi:BPMNLabel>
          <dc:Bounds x="301" y="261" width="90" height="0" />
        </bpmndi:BPMNLabel>
      </bpmndi:BPMNShape>
      <bpmndi:BPMNShape id="ParallelGateway_046yig0_di" bpmnElement="ExclusiveGat
eway_18clflo">
        <dc:Bounds x="498" y="211" width="50" height="50" />
        <bpmndi:BPMNLabel>
          <dc:Bounds x="478" y="261" width="90" height="0" />
        </bpmndi:BPMNLabel>
      </bpmndi:BPMNShape>
      <bpmndi:BPMNEdge id="SequenceFlow_1pm1lpg_di" bpmnElement="SequenceFlow_1pm
1lpg">
        <di:waypoint xsi:type="dc:Point" x="346" y="261" />
        <di:waypoint xsi:type="dc:Point" x="346" y="291" />
        <di:waypoint xsi:type="dc:Point" x="383" y="291" />
        <bpmndi:BPMNLabel>
          <dc:Bounds x="316" y="276" width="90" height="0" />
        </bpmndi:BPMNLabel>
      </bpmndi:BPMNEdge>
      <bpmndi:BPMNEdge id="SequenceFlow_05u8ux5_di" bpmnElement="SequenceFlow_05u
8ux5">
        <di:waypoint xsi:type="dc:Point" x="483" y="291" />
        <di:waypoint xsi:type="dc:Point" x="523" y="291" />
        <di:waypoint xsi:type="dc:Point" x="523" y="261" />
        <bpmndi:BPMNLabel>
          <dc:Bounds x="458" y="276" width="90" height="0" />
        </bpmndi:BPMNLabel>
      </bpmndi:BPMNEdge>
      <bpmndi:BPMNShape id="ServiceTask_1lt0vxh_di" bpmnElement="Task_1n2g1ih">
        <dc:Bounds x="383" y="251" width="100" height="80" />
      </bpmndi:BPMNShape>
      <bpmndi:BPMNEdge id="SequenceFlow_1pxsdl6_di" bpmnElement="SequenceFlow_1px
sdl6">
        <di:waypoint xsi:type="dc:Point" x="936" y="160" />
        <di:waypoint xsi:type="dc:Point" x="961" y="160" />
        <bpmndi:BPMNLabel>
          <dc:Bounds x="903.5" y="145" width="90" height="0" />
        </bpmndi:BPMNLabel>
      </bpmndi:BPMNEdge>
      <bpmndi:BPMNShape id="UserTask_047vhg1_di" bpmnElement="Task_15lfaes">
        <dc:Bounds x="836" y="120" width="100" height="80" />
      </bpmndi:BPMNShape>
      <bpmndi:BPMNShape id="ExclusiveGateway_0ga7p17_di" bpmnElement="ExclusiveGa
teway_0ga7p17" isMarkerVisible="true">
        <dc:Bounds x="961" y="135" width="50" height="50" />
        <bpmndi:BPMNLabel>
          <dc:Bounds x="941" y="185" width="90" height="0" />
        </bpmndi:BPMNLabel>
      </bpmndi:BPMNShape>
      <bpmndi:BPMNEdge id="SequenceFlow_0jiggn5_di" bpmnElement="SequenceFlow_0ji
ggn5">
        <di:waypoint xsi:type="dc:Point" x="1011" y="160" />
        <di:waypoint xsi:type="dc:Point" x="1046" y="160" />
        <bpmndi:BPMNLabel>
          <dc:Bounds x="983.5" y="145" width="90" height="0" />
        </bpmndi:BPMNLabel>
      </bpmndi:BPMNEdge>
      <bpmndi:BPMNEdge id="SequenceFlow_04nl5rk_di" bpmnElement="SequenceFlow_04n
l5rk">
        <di:waypoint xsi:type="dc:Point" x="986" y="185" />
        <di:waypoint xsi:type="dc:Point" x="986" y="239" />
        <di:waypoint xsi:type="dc:Point" x="986" y="239" />
        <di:waypoint xsi:type="dc:Point" x="986" y="284" />
        <bpmndi:BPMNLabel>
          <dc:Bounds x="956" y="239" width="90" height="0" />
        </bpmndi:BPMNLabel>
      </bpmndi:BPMNEdge>
      <bpmndi:BPMNEdge id="SequenceFlow_14exagm_di" bpmnElement="SequenceFlow_14e
xagm">
        <di:waypoint xsi:type="dc:Point" x="810" y="160" />
        <di:waypoint xsi:type="dc:Point" x="836" y="160" />
        <bpmndi:BPMNLabel>
          <dc:Bounds x="778" y="145" width="90" height="0" />
        </bpmndi:BPMNLabel>
      </bpmndi:BPMNEdge>
      <bpmndi:BPMNShape id="IntermediateThrowEvent_1xnsxmp_di" bpmnElement="Inter
mediateThrowEvent_0k5vl3c">
        <dc:Bounds x="774" y="142" width="36" height="36" />
```

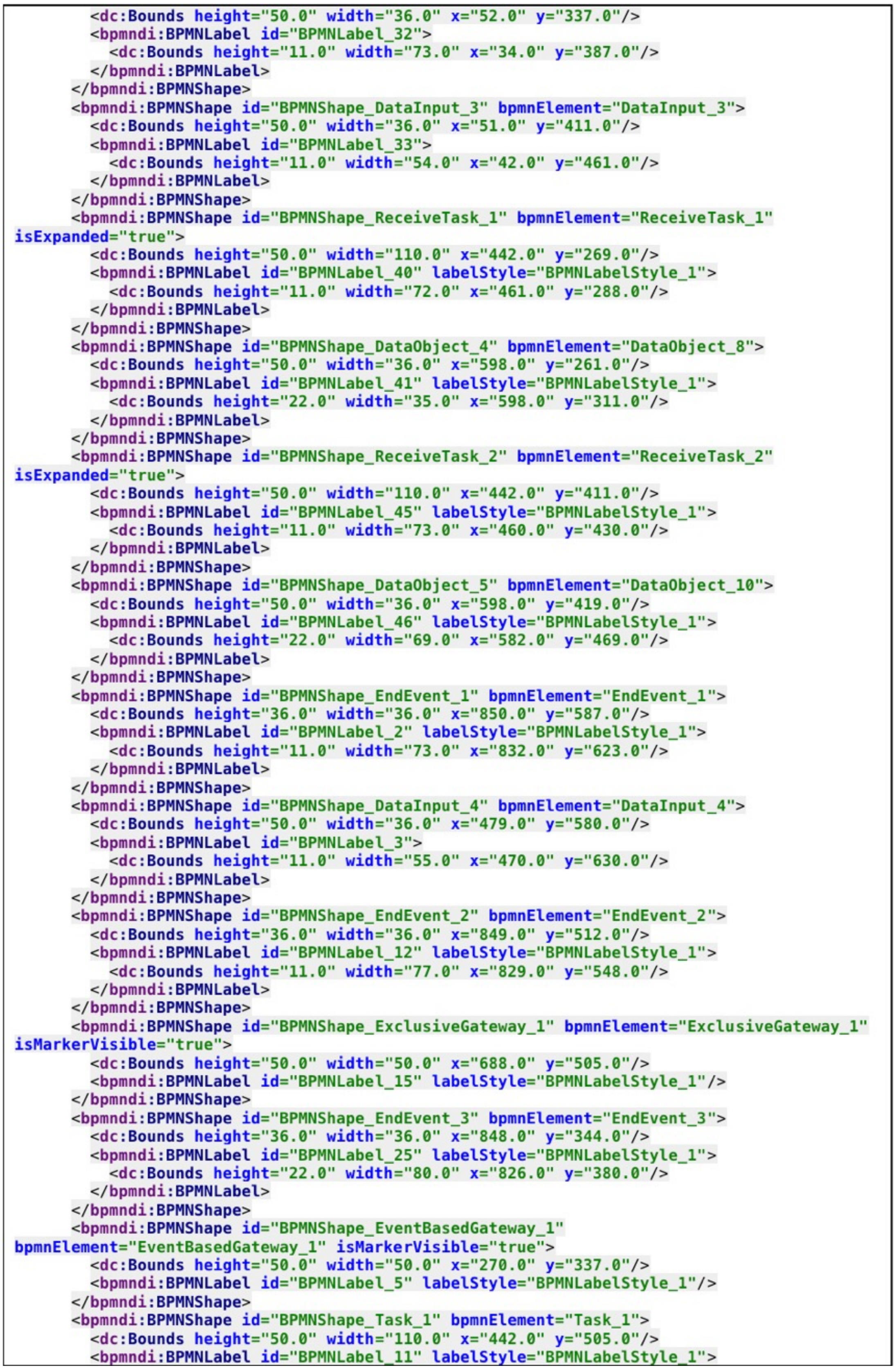
```xml
          <bpmndi:BPMNLabel>
            <dc:Bounds x="747" y="178" width="90" height="25" />
          </bpmndi:BPMNLabel>
        </bpmndi:BPMNShape>
      </bpmndi:BPMNPlane>
    </bpmndi:BPMNDiagram>
</bpmn:definitions>
```

B2. Process Model XML Source for the Loan Assessment Example Using Proposed Approach

```xml
<?xml version="1.0" encoding="UTF-8"?>
<!-- origin at X=0.0 Y=0.0 -->
<bpmn:definitions xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xmlns:bpmn="http://www.omg.org/spec/BPMN/20100524/MODEL"
xmlns:bpmndi="http://www.omg.org/spec/BPMN/20100524/DI"
xmlns:camunda="http://camunda.org/schema/1.0/bpmn"
xmlns:dc="http://www.omg.org/spec/DD/20100524/DC"
xmlns:di="http://www.omg.org/spec/DD/20100524/DI" xmlns:ext="http://org.eclipse.bpmn2/ext"
xmlns:xs="http://www.w3.org/2001/XMLSchema" id="Definitions_1"
exporter="org.eclipse.bpmn2.modeler.core" exporterVersion="1.5.0.Final-v20180515-1642-B1"
targetNamespace="http://bpmn.io/schema/bpmn">
  <bpmn:itemDefinition id="ItemDefinition_19" isCollection="false"
structureRef="solidity:uint"/>
  <bpmn:itemDefinition id="ItemDefinition_1433" isCollection="false"
structureRef="solidity:bool"/>
  <bpmn:message id="Message_1" name="LoanAssess">
    <bpmn:extensionElements>
      <ext:style/>
    </bpmn:extensionElements>
  </bpmn:message>
  <bpmn:message id="Message_2" itemRef="ItemDefinition_19" name="AppraiseProperty">
    <bpmn:extensionElements>
      <ext:style/>
    </bpmn:extensionElements>
  </bpmn:message>
  <bpmn:message id="Message_3" name="LoanAccepted">
    <bpmn:extensionElements>
      <ext:style/>
    </bpmn:extensionElements>
  </bpmn:message>
  <bpmn:message id="Message_4" name="LoanRejected">
    <bpmn:extensionElements>
      <ext:style/>
    </bpmn:extensionElements>
  </bpmn:message>
  <bpmn:message id="Message_5" name="ConfirmationSent">
    <bpmn:extensionElements>
      <ext:style/>
    </bpmn:extensionElements>
  </bpmn:message>
  <bpmn:dataStore id="DataStore_4" name="cost"/>
  <bpmn:dataStore id="DataStore_6" itemSubjectRef="ItemDefinition_19" name="loanAmount"/>
  <bpmn:dataStore id="DataStore_8" itemSubjectRef="ItemDefinition_19"
name="monthlyRevenue"/>
  <bpmn:interface id="Interface_1" implementationRef="json(https://assess.loan.risk.url)"
name="Interface 1">
    <bpmn:operation id="Operation_1" implementationRef=".result" name="Operation 1">
      <bpmn:inMessageRef>Message_1</bpmn:inMessageRef>
    </bpmn:operation>
  </bpmn:interface>
  <bpmn:interface id="Interface_2" implementationRef="json(https://appraise.property.url)"
name="Interface 2">
    <bpmn:operation id="Operation_2" implementationRef=".result" name="Operation 2">
      <bpmn:inMessageRef>Message_2</bpmn:inMessageRef>
    </bpmn:operation>
  </bpmn:interface>
  <bpmn:collaboration id="Collaboration_1" name="Collaboration 1">
```

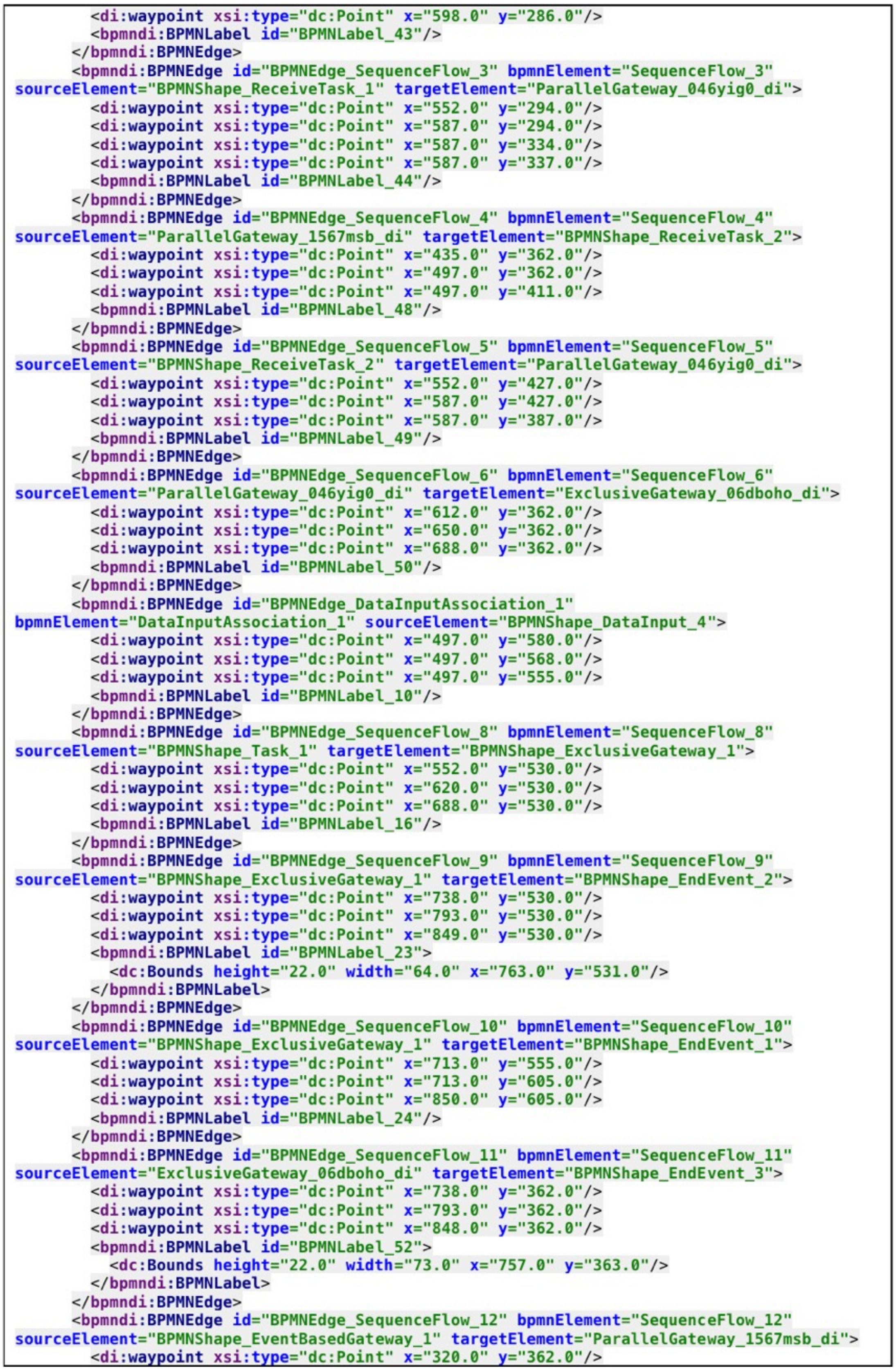
```xml
    <bpmn:participant id="Participant_1" name="Loan Approval" processRef="Process_1"/>
    <bpmn:participant id="Participant_2" name="BPM17_Running_Example Pool"
processRef="BPM17_Running_Example">
      <bpmn:extensionElements>
        <ext:style/>
      </bpmn:extensionElements>
    </bpmn:participant>
  </bpmn:collaboration>
  <bpmn:process id="BPM17_Running_Example" name="BPM17_Running_Example"
isExecutable="false">
    <bpmn:documentation id="Documentation_1">bpmn:documentation>
    <bpmn:ioSpecification id="InputOutputSpecification_11">
      <bpmn:inputSet id="InputSet_11" name="Input Set 11"/>
      <bpmn:outputSet id="OutputSet_11" name="Output Set 11"/>
    </bpmn:ioSpecification>
    <bpmn:association id="Association_4" sourceRef="DataInput_1" targetRef="StartEvent_1"/>
    <bpmn:association id="Association_5" sourceRef="DataInput_2" targetRef="StartEvent_1"/>
    <bpmn:association id="Association_6" sourceRef="DataInput_3" targetRef="StartEvent_1"/>
  </bpmn:process>
  <bpmn:process id="Process_1" name="Loan Approval Process"
definitionalCollaborationRef="Collaboration_1" isExecutable="false">
    <bpmn:ioSpecification id="InputOutputSpecification_2">
      <bpmn:dataInput id="DataInput_4" itemSubjectRef="ItemDefinition_1433"
name="_confirmation"/>
      <bpmn:dataInput id="DataInput_3" itemSubjectRef="ItemDefinition_19"
name="_loanAmount"/>
      <bpmn:dataInput id="DataInput_2" itemSubjectRef="ItemDefinition_19"
name="_monthlyRevenue"/>
      <bpmn:dataInput id="DataInput_1" itemSubjectRef="ItemDefinition_19" name="_cost"/>
      <bpmn:inputSet id="InputSet_2" name="Input Set 2"/>
      <bpmn:outputSet id="OutputSet_2" name="Output Set 2"/>
    </bpmn:ioSpecification>
    <bpmn:dataObject id="DataObject_8" name="loanRisk" itemSubjectRef="ItemDefinition_19">
      <bpmn:dataState id="DataState_5" name="0"/>
    </bpmn:dataObject>
    <bpmn:dataObject id="DataObject_10" name="appraiseProperty"
itemSubjectRef="ItemDefinition_19">
      <bpmn:dataState id="DataState_6" name="0"/>
    </bpmn:dataObject>
    <bpmn:dataStoreReference id="DataStoreReference_3" name="cost"
itemSubjectRef="ItemDefinition_19" dataStoreRef="DataStore_4">
      <bpmn:dataState id="DataState_16"/>
    </bpmn:dataStoreReference>
    <bpmn:dataStoreReference id="DataStoreReference_4" name="loanAmount"
itemSubjectRef="ItemDefinition_19" dataStoreRef="DataStore_6">
      <bpmn:dataState id="DataState_17"/>
    </bpmn:dataStoreReference>
    <bpmn:dataStoreReference id="DataStoreReference_5" name="monthlyRevenue"
itemSubjectRef="ItemDefinition_19" dataStoreRef="DataStore_8">
      <bpmn:dataState id="DataState_18"/>
    </bpmn:dataStoreReference>
    <bpmn:endEvent id="EndEvent_3" name="Confirmation event sent">
      <bpmn:incoming>SequenceFlow_11</bpmn:incoming>
      <bpmn:dataInput id="DataInput_9" name="Message_6_Input"/>
      <bpmn:dataInputAssociation id="DataInputAssociation_5">
        <bpmn:targetRef>DataInput_9</bpmn:targetRef>
      </bpmn:dataInputAssociation>
      <bpmn:inputSet id="InputSet_6" name="Input Set 6">
        <bpmn:dataInputRefs>DataInput_9</bpmn:dataInputRefs>
      </bpmn:inputSet>
      <bpmn:messageEventDefinition id="MessageEventDefinition_6" messageRef="Message_5"/>
    </bpmn:endEvent>
    <bpmn:parallelGateway id="ExclusiveGateway_0o4nv8y" gatewayDirection="Diverging">
      <bpmn:incoming>SequenceFlow_12</bpmn:incoming>
      <bpmn:outgoing>SequenceFlow_2</bpmn:outgoing>
      <bpmn:outgoing>SequenceFlow_4</bpmn:outgoing>
    </bpmn:parallelGateway>
    <bpmn:sequenceFlow id="SequenceFlow_2" sourceRef="ExclusiveGateway_0o4nv8y"
targetRef="ReceiveTask_1"/>
    <bpmn:sequenceFlow id="SequenceFlow_4" sourceRef="ExclusiveGateway_0o4nv8y"
targetRef="ReceiveTask_2"/>
    <bpmn:endEvent id="EndEvent_1ubxmre" name="Loan app rejected">
      <bpmn:incoming>SequenceFlow_0vbi21y</bpmn:incoming>
      <bpmn:dataInput id="DataInput_7" name="Message_2_Input"/>
      <bpmn:dataInputAssociation id="DataInputAssociation_3">
        <bpmn:targetRef>DataInput_7</bpmn:targetRef>
      </bpmn:dataInputAssociation>
      <bpmn:inputSet id="InputSet_3" name="Input Set 3">
        <bpmn:dataInputRefs>DataInput_7</bpmn:dataInputRefs>
      </bpmn:inputSet>
      <bpmn:messageEventDefinition id="MessageEventDefinition_2" messageRef="Message_4"/>
    </bpmn:endEvent>
    <bpmn:exclusiveGateway id="ExclusiveGateway_1" gatewayDirection="Diverging">
      <bpmn:incoming>SequenceFlow_8</bpmn:incoming>
      <bpmn:outgoing>SequenceFlow_9</bpmn:outgoing>
      <bpmn:outgoing>SequenceFlow_10</bpmn:outgoing>
    </bpmn:exclusiveGateway>
    <bpmn:sequenceFlow id="SequenceFlow_9" name="_confirmation == true"
sourceRef="ExclusiveGateway_1" targetRef="EndEvent_2">
      <bpmn:conditionExpression xsi:type="bpmn:tFormalExpression"
id="FormalExpression_3">_confirmation == true</bpmn:conditionExpression>
    </bpmn:sequenceFlow>
    <bpmn:sequenceFlow id="SequenceFlow_10" sourceRef="ExclusiveGateway_1"
targetRef="EndEvent_1"/>
    <bpmn:task id="Task_1" name="Confirm Acceptance ">
      <bpmn:incoming>SequenceFlow_13</bpmn:incoming>
      <bpmn:outgoing>SequenceFlow_8</bpmn:outgoing>
      <bpmn:ioSpecification id="InputOutputSpecification_4">
        <bpmn:dataInput id="DataInput_5" itemSubjectRef="ItemDefinition_1433"
name="_confirmation"/>
        <bpmn:inputSet id="InputSet_4" name="Input Set 4">
          <bpmn:dataInputRefs>DataInput_5</bpmn:dataInputRefs>
        </bpmn:inputSet>
        <bpmn:outputSet id="OutputSet_4" name="Output Set 4"/>
      </bpmn:ioSpecification>
      <bpmn:dataInputAssociation id="DataInputAssociation_1">
        <bpmn:sourceRef>DataInput_4</bpmn:sourceRef>
        <bpmn:targetRef>DataInput_5</bpmn:targetRef>
      </bpmn:dataInputAssociation>
    </bpmn:task>
    <bpmn:sequenceFlow id="SequenceFlow_8" sourceRef="Task_1"
targetRef="ExclusiveGateway_1"/>
    <bpmn:eventBasedGateway id="EventBasedGateway_1" gatewayDirection="Diverging">
      <bpmn:incoming>SequenceFlow_14</bpmn:incoming>
      <bpmn:outgoing>SequenceFlow_12</bpmn:outgoing>
      <bpmn:outgoing>SequenceFlow_13</bpmn:outgoing>
    </bpmn:eventBasedGateway>
    <bpmn:sequenceFlow id="SequenceFlow_12" sourceRef="EventBasedGateway_1"
targetRef="ExclusiveGateway_0o4nv8y"/>
    <bpmn:sequenceFlow id="SequenceFlow_13" sourceRef="EventBasedGateway_1"
targetRef="Task_1"/>
    <bpmn:exclusiveGateway id="ExclusiveGateway_06dboho" gatewayDirection="Diverging">
      <bpmn:incoming>SequenceFlow_6</bpmn:incoming>
      <bpmn:outgoing>SequenceFlow_0vbi21y</bpmn:outgoing>
      <bpmn:outgoing>SequenceFlow_11</bpmn:outgoing>
    </bpmn:exclusiveGateway>
    <bpmn:sequenceFlow id="SequenceFlow_0vbi21y" sourceRef="ExclusiveGateway_06dboho"
targetRef="EndEvent_1ubxmre"/>
    <bpmn:sequenceFlow id="SequenceFlow_11" name="loanRisk >= appraiseProperty"
sourceRef="ExclusiveGateway_06dboho" targetRef="EndEvent_3">
      <bpmn:conditionExpression xsi:type="bpmn:tFormalExpression"
id="FormalExpression_1">loanRisk >= appraiseProperty</bpmn:conditionExpression>
    </bpmn:sequenceFlow>
    <bpmn:receiveTask id="ReceiveTask_1" name="Assess Loan Risk" messageRef="Message_1"
operationRef="Operation_1">
      <bpmn:incoming>SequenceFlow_2</bpmn:incoming>
      <bpmn:outgoing>SequenceFlow_3</bpmn:outgoing>
      <bpmn:ioSpecification id="InputOutputSpecification_24">
        <bpmn:dataInput id="DataInput_22" itemSubjectRef="ItemDefinition_19"
name="monthlyRevenue"/>
        <bpmn:dataInput id="DataInput_23" itemSubjectRef="ItemDefinition_19"
name="loanAmount"/>
        <bpmn:dataOutput id="DataOutput_7" itemSubjectRef="ItemDefinition_19"
name="output1"/>
        <bpmn:inputSet id="InputSet_24" name="Input Set 24">
          <bpmn:dataInputRefs>DataInput_22</bpmn:dataInputRefs>
          <bpmn:dataInputRefs>DataInput_23</bpmn:dataInputRefs>
        </bpmn:inputSet>
        <bpmn:outputSet id="OutputSet_25" name="Output Set 25">
          <bpmn:dataOutputRefs>DataOutput_7</bpmn:dataOutputRefs>
        </bpmn:outputSet>
      </bpmn:ioSpecification>
      <bpmn:dataInputAssociation id="DataInputAssociation_15">
        <bpmn:sourceRef>DataStoreReference_5</bpmn:sourceRef>
        <bpmn:targetRef>DataInput_22</bpmn:targetRef>
      </bpmn:dataInputAssociation>
      <bpmn:dataInputAssociation id="DataInputAssociation_16">
        <bpmn:sourceRef>DataStoreReference_4</bpmn:sourceRef>
        <bpmn:targetRef>DataInput_23</bpmn:targetRef>
      </bpmn:dataInputAssociation>
      <bpmn:dataOutputAssociation id="DataOutputAssociation_7">
        <bpmn:sourceRef>DataOutput_7</bpmn:sourceRef>
        <bpmn:targetRef>DataObject_8</bpmn:targetRef>
      </bpmn:dataOutputAssociation>
    </bpmn:receiveTask>
    <bpmn:sequenceFlow id="SequenceFlow_3" sourceRef="ReceiveTask_1"
targetRef="ExclusiveGateway_18clflo"/>
    <bpmn:endEvent id="EndEvent_1" name="Loan app rejected">
      <bpmn:incoming>SequenceFlow_10</bpmn:incoming>
      <bpmn:dataInput id="DataInput_8" name="Message_4_Input"/>
      <bpmn:dataInputAssociation id="DataInputAssociation_4">
        <bpmn:targetRef>DataInput_8</bpmn:targetRef>
      </bpmn:dataInputAssociation>
      <bpmn:inputSet id="InputSet_5" name="Input Set 5">
        <bpmn:dataInputRefs>DataInput_8</bpmn:dataInputRefs>
      </bpmn:inputSet>
      <bpmn:messageEventDefinition id="MessageEventDefinition_4" messageRef="Message_4"/>
    </bpmn:endEvent>
    <bpmn:endEvent id="EndEvent_2" name="Loan app accepted">
      <bpmn:incoming>SequenceFlow_9</bpmn:incoming>
      <bpmn:dataInput id="DataInput_6" name="Message_5_Input"/>
      <bpmn:dataInputAssociation id="DataInputAssociation_2">
        <bpmn:targetRef>DataInput_6</bpmn:targetRef>
      </bpmn:dataInputAssociation>
      <bpmn:inputSet id="InputSet_1" name="Input Set 1">
        <bpmn:dataInputRefs>DataInput_6</bpmn:dataInputRefs>
      </bpmn:inputSet>
      <bpmn:messageEventDefinition id="MessageEventDefinition_5" messageRef="Message_3"/>
    </bpmn:endEvent>
    <bpmn:receiveTask id="ReceiveTask_2" name="Appraise Prpoerty" messageRef="Message_2"
operationRef="Operation_2">
      <bpmn:incoming>SequenceFlow_4</bpmn:incoming>
      <bpmn:outgoing>SequenceFlow_5</bpmn:outgoing>
      <bpmn:ioSpecification id="InputOutputSpecification_34">
        <bpmn:dataInput id="DataInput_24" itemSubjectRef="ItemDefinition_19" name="cost"/>
        <bpmn:dataOutput id="DataOutput_8" itemSubjectRef="ItemDefinition_19"
name="output1"/>
        <bpmn:dataOutput id="DataOutput_9" itemSubjectRef="ItemDefinition_19"
name="output2"/>
        <bpmn:inputSet id="InputSet_34" name="Input Set 34">
          <bpmn:dataInputRefs>DataInput_24</bpmn:dataInputRefs>
        </bpmn:inputSet>
        <bpmn:outputSet id="OutputSet_35" name="Output Set 35">
          <bpmn:dataOutputRefs>DataOutput_8</bpmn:dataOutputRefs>
          <bpmn:dataOutputRefs>DataOutput_9</bpmn:dataOutputRefs>
        </bpmn:outputSet>
      </bpmn:ioSpecification>
      <bpmn:dataInputAssociation id="DataInputAssociation_17">
        <bpmn:sourceRef>DataStoreReference_3</bpmn:sourceRef>
        <bpmn:targetRef>DataInput_24</bpmn:targetRef>
      </bpmn:dataInputAssociation>
      <bpmn:dataOutputAssociation id="DataOutputAssociation_6">
        <bpmn:sourceRef>DataOutput_8</bpmn:sourceRef>
        <bpmn:targetRef>DataObject_10</bpmn:targetRef>
      </bpmn:dataOutputAssociation>
    </bpmn:receiveTask>
    <bpmn:sequenceFlow id="SequenceFlow_5" sourceRef="ReceiveTask_2"
targetRef="ExclusiveGateway_18clflo"/>
    <bpmn:startEvent id="StartEvent_1" name="Start">
      <bpmn:outgoing>SequenceFlow_14</bpmn:outgoing>
      <bpmn:dataOutput id="DataOutput_1" itemSubjectRef="ItemDefinition_19" name="_cost"/>
      <bpmn:dataOutput id="DataOutput_2" itemSubjectRef="ItemDefinition_19"
```

```
name="_loanAmount"/>
      <bpmn:dataOutput id="DataOutput_3" itemSubjectRef="ItemDefinition_19"
name="_monthlyRevenue"/>
      <bpmn:dataOutputAssociation id="DataOutputAssociation_2">
        <bpmn:sourceRef>DataOutput_2</bpmn:sourceRef>
        <bpmn:targetRef>DataStoreReference_4</bpmn:targetRef>
      </bpmn:dataOutputAssociation>
      <bpmn:dataOutputAssociation id="DataOutputAssociation_23">
        <bpmn:sourceRef>DataOutput_1</bpmn:sourceRef>
        <bpmn:targetRef>DataStoreReference_3</bpmn:targetRef>
      </bpmn:dataOutputAssociation>
      <bpmn:dataOutputAssociation id="DataOutputAssociation_24">
        <bpmn:sourceRef>DataOutput_3</bpmn:sourceRef>
        <bpmn:targetRef>DataStoreReference_5</bpmn:targetRef>
      </bpmn:dataOutputAssociation>
      <bpmn:outputSet id="OutputSet_12" name="Output Set 12">
        <bpmn:dataOutputRefs>DataOutput_1</bpmn:dataOutputRefs>
        <bpmn:dataOutputRefs>DataOutput_2</bpmn:dataOutputRefs>
        <bpmn:dataOutputRefs>DataOutput_3</bpmn:dataOutputRefs>
      </bpmn:outputSet>
    </bpmn:startEvent>
    <bpmn:sequenceFlow id="SequenceFlow_14" sourceRef="StartEvent_1"
targetRef="EventBasedGateway_1"/>
    <bpmn:parallelGateway id="ExclusiveGateway_18clflo" gatewayDirection="Converging">
      <bpmn:incoming>SequenceFlow_3</bpmn:incoming>
      <bpmn:incoming>SequenceFlow_5</bpmn:incoming>
      <bpmn:outgoing>SequenceFlow_6</bpmn:outgoing>
    </bpmn:parallelGateway>
    <bpmn:sequenceFlow id="SequenceFlow_6" sourceRef="ExclusiveGateway_18clflo"
targetRef="ExclusiveGateway_06dboho"/>
  </bpmn:process>
  <bpmndi:BPMNDiagram id="BPMNDiagram_1">
    <bpmndi:BPMNPlane id="BPMNPlane_1" bpmnElement="Collaboration_1">
      <bpmndi:BPMNShape id="BPMNShape_Participant_1" bpmnElement="Participant_1"
isHorizontal="true">
        <dc:Bounds height="491.0" width="929.0" x="0.0" y="160.0"/>
        <bpmndi:BPMNLabel id="BPMNLabel_51" labelStyle="BPMNLabelStyle_1">
          <dc:Bounds height="58.0" width="11.0" x="6.0" y="376.0"/>
        </bpmndi:BPMNLabel>
      </bpmndi:BPMNShape>
      <bpmndi:BPMNShape id="_BPMNShape_StartEvent_2" bpmnElement="StartEvent_1">
        <dc:Bounds height="36.0" width="36.0" x="137.0" y="344.0"/>
        <bpmndi:BPMNLabel id="BPMNLabel_1" labelStyle="BPMNLabelStyle_1">
          <dc:Bounds height="11.0" width="20.0" x="145.0" y="380.0"/>
        </bpmndi:BPMNLabel>
      </bpmndi:BPMNShape>
      <bpmndi:BPMNShape id="ExclusiveGateway_06dboho_di"
bpmnElement="ExclusiveGateway_06dboho" isMarkerVisible="true">
        <dc:Bounds height="50.0" width="50.0" x="688.0" y="337.0"/>
        <bpmndi:BPMNLabel id="BPMNLabel_4" labelStyle="BPMNLabelStyle_1"/>
      </bpmndi:BPMNShape>
      <bpmndi:BPMNShape id="EndEvent_1hyovvg_di" bpmnElement="EndEvent_1ubxmre">
        <dc:Bounds height="36.0" width="36.0" x="848.0" y="411.0"/>
        <bpmndi:BPMNLabel id="BPMNLabel_9" labelStyle="BPMNLabelStyle_1">
          <dc:Bounds height="11.0" width="73.0" x="830.0" y="447.0"/>
        </bpmndi:BPMNLabel>
      </bpmndi:BPMNShape>
      <bpmndi:BPMNShape id="ParallelGateway_1567msb_di"
bpmnElement="ExclusiveGateway_0o4nv8y" isMarkerVisible="true">
        <dc:Bounds height="50.0" width="50.0" x="385.0" y="337.0"/>
        <bpmndi:BPMNLabel id="BPMNLabel_13" labelStyle="BPMNLabelStyle_1"/>
      </bpmndi:BPMNShape>
      <bpmndi:BPMNShape id="ParallelGateway_046yig0_di"
bpmnElement="ExclusiveGateway_18clflo" isMarkerVisible="true">
        <dc:Bounds height="50.0" width="50.0" x="562.0" y="337.0"/>
        <bpmndi:BPMNLabel id="BPMNLabel_14" labelStyle="BPMNLabelStyle_1"/>
      </bpmndi:BPMNShape>
      <bpmndi:BPMNShape id="BPMNShape_DataInput_1" bpmnElement="DataInput_1">
        <dc:Bounds height="50.0" width="36.0" x="51.0" y="262.0"/>
        <bpmndi:BPMNLabel id="BPMNLabel_31">
          <dc:Bounds height="11.0" width="22.0" x="58.0" y="312.0"/>
        </bpmndi:BPMNLabel>
      </bpmndi:BPMNShape>
      <bpmndi:BPMNShape id="BPMNShape_DataInput_2" bpmnElement="DataInput_2">
```



```
        <dc:Bounds height="50.0" width="36.0" x="52.0" y="337.0"/>
        <bpmndi:BPMNLabel id="BPMNLabel_32">
          <dc:Bounds height="11.0" width="73.0" x="34.0" y="387.0"/>
        </bpmndi:BPMNLabel>
      </bpmndi:BPMNShape>
      <bpmndi:BPMNShape id="BPMNShape_DataInput_3" bpmnElement="DataInput_3">
        <dc:Bounds height="50.0" width="36.0" x="51.0" y="411.0"/>
        <bpmndi:BPMNLabel id="BPMNLabel_33">
          <dc:Bounds height="11.0" width="54.0" x="42.0" y="461.0"/>
        </bpmndi:BPMNLabel>
      </bpmndi:BPMNShape>
      <bpmndi:BPMNShape id="BPMNShape_ReceiveTask_1" bpmnElement="ReceiveTask_1"
isExpanded="true">
        <dc:Bounds height="50.0" width="110.0" x="442.0" y="269.0"/>
        <bpmndi:BPMNLabel id="BPMNLabel_40" labelStyle="BPMNLabelStyle_1">
          <dc:Bounds height="11.0" width="72.0" x="461.0" y="288.0"/>
        </bpmndi:BPMNLabel>
      </bpmndi:BPMNShape>
      <bpmndi:BPMNShape id="BPMNShape_DataObject_4" bpmnElement="DataObject_8">
        <dc:Bounds height="50.0" width="36.0" x="598.0" y="261.0"/>
        <bpmndi:BPMNLabel id="BPMNLabel_41" labelStyle="BPMNLabelStyle_1">
          <dc:Bounds height="22.0" width="35.0" x="598.0" y="311.0"/>
        </bpmndi:BPMNLabel>
      </bpmndi:BPMNShape>
      <bpmndi:BPMNShape id="BPMNShape_ReceiveTask_2" bpmnElement="ReceiveTask_2"
isExpanded="true">
        <dc:Bounds height="50.0" width="110.0" x="442.0" y="411.0"/>
        <bpmndi:BPMNLabel id="BPMNLabel_45" labelStyle="BPMNLabelStyle_1">
          <dc:Bounds height="11.0" width="73.0" x="460.0" y="430.0"/>
        </bpmndi:BPMNLabel>
      </bpmndi:BPMNShape>
      <bpmndi:BPMNShape id="BPMNShape_DataObject_5" bpmnElement="DataObject_10">
        <dc:Bounds height="50.0" width="36.0" x="598.0" y="419.0"/>
        <bpmndi:BPMNLabel id="BPMNLabel_46" labelStyle="BPMNLabelStyle_1">
          <dc:Bounds height="22.0" width="69.0" x="582.0" y="469.0"/>
        </bpmndi:BPMNLabel>
      </bpmndi:BPMNShape>
      <bpmndi:BPMNShape id="BPMNShape_EndEvent_1" bpmnElement="EndEvent_1">
        <dc:Bounds height="36.0" width="36.0" x="850.0" y="587.0"/>
        <bpmndi:BPMNLabel id="BPMNLabel_2" labelStyle="BPMNLabelStyle_1">
          <dc:Bounds height="11.0" width="73.0" x="832.0" y="623.0"/>
        </bpmndi:BPMNLabel>
      </bpmndi:BPMNShape>
      <bpmndi:BPMNShape id="BPMNShape_DataInput_4" bpmnElement="DataInput_4">
        <dc:Bounds height="50.0" width="36.0" x="479.0" y="580.0"/>
        <bpmndi:BPMNLabel id="BPMNLabel_3">
          <dc:Bounds height="11.0" width="55.0" x="470.0" y="630.0"/>
        </bpmndi:BPMNLabel>
      </bpmndi:BPMNShape>
      <bpmndi:BPMNShape id="BPMNShape_EndEvent_2" bpmnElement="EndEvent_2">
        <dc:Bounds height="36.0" width="36.0" x="849.0" y="512.0"/>
        <bpmndi:BPMNLabel id="BPMNLabel_12" labelStyle="BPMNLabelStyle_1">
          <dc:Bounds height="11.0" width="77.0" x="829.0" y="548.0"/>
        </bpmndi:BPMNLabel>
      </bpmndi:BPMNShape>
      <bpmndi:BPMNShape id="BPMNShape_ExclusiveGateway_1" bpmnElement="ExclusiveGateway_1"
isMarkerVisible="true">
        <dc:Bounds height="50.0" width="50.0" x="688.0" y="505.0"/>
        <bpmndi:BPMNLabel id="BPMNLabel_15" labelStyle="BPMNLabelStyle_1"/>
      </bpmndi:BPMNShape>
      <bpmndi:BPMNShape id="BPMNShape_EndEvent_3" bpmnElement="EndEvent_3">
        <dc:Bounds height="36.0" width="36.0" x="848.0" y="344.0"/>
        <bpmndi:BPMNLabel id="BPMNLabel_25" labelStyle="BPMNLabelStyle_1">
          <dc:Bounds height="22.0" width="80.0" x="826.0" y="380.0"/>
        </bpmndi:BPMNLabel>
      </bpmndi:BPMNShape>
      <bpmndi:BPMNShape id="BPMNShape_EventBasedGateway_1"
bpmnElement="EventBasedGateway_1" isMarkerVisible="true">
        <dc:Bounds height="50.0" width="50.0" x="270.0" y="337.0"/>
        <bpmndi:BPMNLabel id="BPMNLabel_5" labelStyle="BPMNLabelStyle_1"/>
      </bpmndi:BPMNShape>
      <bpmndi:BPMNShape id="BPMNShape_Task_1" bpmnElement="Task_1">
        <dc:Bounds height="50.0" width="110.0" x="442.0" y="505.0"/>
        <bpmndi:BPMNLabel id="BPMNLabel_11" labelStyle="BPMNLabelStyle_1">
```

```
            <dc:Bounds height="11.0" width="84.0" x="455.0" y="524.0"/>
          </bpmndi:BPMNLabel>
        </bpmndi:BPMNShape>
        <bpmndi:BPMNShape id="BPMNShape_DataStoreReference_3"
bpmnElement="DataStoreReference_3">
          <dc:Bounds height="50.0" width="50.0" x="230.0" y="411.0"/>
          <bpmndi:BPMNLabel id="BPMNLabel_26" labelStyle="BPMNLabelStyle_1">
            <dc:Bounds height="11.0" width="17.0" x="246.0" y="461.0"/>
          </bpmndi:BPMNLabel>
        </bpmndi:BPMNShape>
        <bpmndi:BPMNShape id="BPMNShape_DataStoreReference_4"
bpmnElement="DataStoreReference_4">
          <dc:Bounds height="50.0" width="50.0" x="230.0" y="261.0"/>
          <bpmndi:BPMNLabel id="BPMNLabel_47" labelStyle="BPMNLabelStyle_1">
            <dc:Bounds height="11.0" width="49.0" x="230.0" y="311.0"/>
          </bpmndi:BPMNLabel>
        </bpmndi:BPMNShape>
        <bpmndi:BPMNShape id="BPMNShape_DataStoreReference_5"
bpmnElement="DataStoreReference_5">
          <dc:Bounds height="50.0" width="50.0" x="230.0" y="180.0"/>
          <bpmndi:BPMNLabel id="BPMNLabel_53" labelStyle="BPMNLabelStyle_1">
            <dc:Bounds height="11.0" width="68.0" x="221.0" y="230.0"/>
          </bpmndi:BPMNLabel>
        </bpmndi:BPMNShape>
        <bpmndi:BPMNEdge id="SequenceFlow_0vbi21y_di" bpmnElement="SequenceFlow_0vbi21y"
sourceElement="ExclusiveGateway_06dboho_di" targetElement="EndEvent_1hyovvg_di">
          <di:waypoint xsi:type="dc:Point" x="713.0" y="387.0"/>
          <di:waypoint xsi:type="dc:Point" x="713.0" y="429.0"/>
          <di:waypoint xsi:type="dc:Point" x="848.0" y="429.0"/>
          <bpmndi:BPMNLabel id="BPMNLabel_8"/>
        </bpmndi:BPMNEdge>
        <bpmndi:BPMNEdge id="BPMNEdge_Association_4" bpmnElement="Association_4"
sourceElement="BPMNShape_DataInput_1" targetElement="_BPMNShape_StartEvent_2">
          <di:waypoint xsi:type="dc:Point" x="87.0" y="287.0"/>
          <di:waypoint xsi:type="dc:Point" x="112.0" y="287.0"/>
          <di:waypoint xsi:type="dc:Point" x="112.0" y="362.0"/>
          <di:waypoint xsi:type="dc:Point" x="137.0" y="362.0"/>
          <bpmndi:BPMNLabel id="BPMNLabel_34"/>
        </bpmndi:BPMNEdge>
        <bpmndi:BPMNEdge id="BPMNEdge_Association_5" bpmnElement="Association_5"
sourceElement="BPMNShape_DataInput_2" targetElement="_BPMNShape_StartEvent_2">
          <di:waypoint xsi:type="dc:Point" x="88.0" y="362.0"/>
          <di:waypoint xsi:type="dc:Point" x="112.0" y="362.0"/>
          <di:waypoint xsi:type="dc:Point" x="137.0" y="362.0"/>
          <bpmndi:BPMNLabel id="BPMNLabel_35"/>
        </bpmndi:BPMNEdge>
        <bpmndi:BPMNEdge id="BPMNEdge_Association_6" bpmnElement="Association_6"
sourceElement="BPMNShape_DataInput_3" targetElement="_BPMNShape_StartEvent_2">
          <di:waypoint xsi:type="dc:Point" x="87.0" y="436.0"/>
          <di:waypoint xsi:type="dc:Point" x="128.0" y="436.0"/>
          <di:waypoint xsi:type="dc:Point" x="128.0" y="362.0"/>
          <di:waypoint xsi:type="dc:Point" x="137.0" y="362.0"/>
          <bpmndi:BPMNLabel id="BPMNLabel_36"/>
        </bpmndi:BPMNEdge>
        <bpmndi:BPMNEdge id="BPMNEdge_DataOutputAssociation_2"
bpmnElement="DataOutputAssociation_2" sourceElement="_BPMNShape_StartEvent_2">
          <di:waypoint xsi:type="dc:Point" x="155.0" y="344.0"/>
          <di:waypoint xsi:type="dc:Point" x="155.0" y="286.0"/>
          <di:waypoint xsi:type="dc:Point" x="230.0" y="286.0"/>
          <bpmndi:BPMNLabel id="BPMNLabel_38"/>
        </bpmndi:BPMNEdge>
        <bpmndi:BPMNEdge id="BPMNEdge_SequenceFlow_2" bpmnElement="SequenceFlow_2"
sourceElement="ParallelGateway_1567msb_di" targetElement="BPMNShape_ReceiveTask_1">
          <di:waypoint xsi:type="dc:Point" x="410.0" y="337.0"/>
          <di:waypoint xsi:type="dc:Point" x="410.0" y="306.0"/>
          <di:waypoint xsi:type="dc:Point" x="442.0" y="306.0"/>
          <bpmndi:BPMNLabel id="BPMNLabel_42"/>
        </bpmndi:BPMNEdge>
        <bpmndi:BPMNEdge id="BPMNEdge_DataOutputAssociation_4"
bpmnElement="DataOutputAssociation_7" sourceElement="BPMNShape_ReceiveTask_1"
targetElement="BPMNShape_DataObject_4">
          <di:waypoint xsi:type="dc:Point" x="552.0" y="285.0"/>
          <di:waypoint xsi:type="dc:Point" x="585.0" y="285.0"/>
          <di:waypoint xsi:type="dc:Point" x="585.0" y="286.0"/>
```



```xml
        <di:waypoint xsi:type="dc:Point" x="598.0" y="286.0"/>
        <bpmndi:BPMNLabel id="BPMNLabel_43"/>
      </bpmndi:BPMNEdge>
      <bpmndi:BPMNEdge id="BPMNEdge_SequenceFlow_3" bpmnElement="SequenceFlow_3"
sourceElement="BPMNShape_ReceiveTask_1" targetElement="ParallelGateway_046yig0_di">
        <di:waypoint xsi:type="dc:Point" x="552.0" y="294.0"/>
        <di:waypoint xsi:type="dc:Point" x="587.0" y="294.0"/>
        <di:waypoint xsi:type="dc:Point" x="587.0" y="334.0"/>
        <di:waypoint xsi:type="dc:Point" x="587.0" y="337.0"/>
        <bpmndi:BPMNLabel id="BPMNLabel_44"/>
      </bpmndi:BPMNEdge>
      <bpmndi:BPMNEdge id="BPMNEdge_SequenceFlow_4" bpmnElement="SequenceFlow_4"
sourceElement="ParallelGateway_1567msb_di" targetElement="BPMNShape_ReceiveTask_2">
        <di:waypoint xsi:type="dc:Point" x="435.0" y="362.0"/>
        <di:waypoint xsi:type="dc:Point" x="497.0" y="362.0"/>
        <di:waypoint xsi:type="dc:Point" x="497.0" y="411.0"/>
        <bpmndi:BPMNLabel id="BPMNLabel_48"/>
      </bpmndi:BPMNEdge>
      <bpmndi:BPMNEdge id="BPMNEdge_SequenceFlow_5" bpmnElement="SequenceFlow_5"
sourceElement="BPMNShape_ReceiveTask_2" targetElement="ParallelGateway_046yig0_di">
        <di:waypoint xsi:type="dc:Point" x="552.0" y="427.0"/>
        <di:waypoint xsi:type="dc:Point" x="587.0" y="427.0"/>
        <di:waypoint xsi:type="dc:Point" x="587.0" y="387.0"/>
        <bpmndi:BPMNLabel id="BPMNLabel_49"/>
      </bpmndi:BPMNEdge>
      <bpmndi:BPMNEdge id="BPMNEdge_SequenceFlow_6" bpmnElement="SequenceFlow_6"
sourceElement="ParallelGateway_046yig0_di" targetElement="ExclusiveGateway_06dboho_di">
        <di:waypoint xsi:type="dc:Point" x="612.0" y="362.0"/>
        <di:waypoint xsi:type="dc:Point" x="650.0" y="362.0"/>
        <di:waypoint xsi:type="dc:Point" x="688.0" y="362.0"/>
        <bpmndi:BPMNLabel id="BPMNLabel_50"/>
      </bpmndi:BPMNEdge>
      <bpmndi:BPMNEdge id="BPMNEdge_DataInputAssociation_1"
bpmnElement="DataInputAssociation_1" sourceElement="BPMNShape_DataInput_4">
        <di:waypoint xsi:type="dc:Point" x="497.0" y="580.0"/>
        <di:waypoint xsi:type="dc:Point" x="497.0" y="568.0"/>
        <di:waypoint xsi:type="dc:Point" x="497.0" y="555.0"/>
        <bpmndi:BPMNLabel id="BPMNLabel_10"/>
      </bpmndi:BPMNEdge>
      <bpmndi:BPMNEdge id="BPMNEdge_SequenceFlow_8" bpmnElement="SequenceFlow_8"
sourceElement="BPMNShape_Task_1" targetElement="BPMNShape_ExclusiveGateway_1">
        <di:waypoint xsi:type="dc:Point" x="552.0" y="530.0"/>
        <di:waypoint xsi:type="dc:Point" x="620.0" y="530.0"/>
        <di:waypoint xsi:type="dc:Point" x="688.0" y="530.0"/>
        <bpmndi:BPMNLabel id="BPMNLabel_16"/>
      </bpmndi:BPMNEdge>
      <bpmndi:BPMNEdge id="BPMNEdge_SequenceFlow_9" bpmnElement="SequenceFlow_9"
sourceElement="BPMNShape_ExclusiveGateway_1" targetElement="BPMNShape_EndEvent_2">
        <di:waypoint xsi:type="dc:Point" x="738.0" y="530.0"/>
        <di:waypoint xsi:type="dc:Point" x="793.0" y="530.0"/>
        <di:waypoint xsi:type="dc:Point" x="849.0" y="530.0"/>
        <bpmndi:BPMNLabel id="BPMNLabel_23">
          <dc:Bounds height="22.0" width="64.0" x="763.0" y="531.0"/>
        </bpmndi:BPMNLabel>
      </bpmndi:BPMNEdge>
      <bpmndi:BPMNEdge id="BPMNEdge_SequenceFlow_10" bpmnElement="SequenceFlow_10"
sourceElement="BPMNShape_ExclusiveGateway_1" targetElement="BPMNShape_EndEvent_1">
        <di:waypoint xsi:type="dc:Point" x="713.0" y="555.0"/>
        <di:waypoint xsi:type="dc:Point" x="713.0" y="605.0"/>
        <di:waypoint xsi:type="dc:Point" x="850.0" y="605.0"/>
        <bpmndi:BPMNLabel id="BPMNLabel_24"/>
      </bpmndi:BPMNEdge>
      <bpmndi:BPMNEdge id="BPMNEdge_SequenceFlow_11" bpmnElement="SequenceFlow_11"
sourceElement="ExclusiveGateway_06dboho_di" targetElement="BPMNShape_EndEvent_3">
        <di:waypoint xsi:type="dc:Point" x="738.0" y="362.0"/>
        <di:waypoint xsi:type="dc:Point" x="793.0" y="362.0"/>
        <di:waypoint xsi:type="dc:Point" x="848.0" y="362.0"/>
        <bpmndi:BPMNLabel id="BPMNLabel_52">
          <dc:Bounds height="22.0" width="73.0" x="757.0" y="363.0"/>
        </bpmndi:BPMNLabel>
      </bpmndi:BPMNEdge>
      <bpmndi:BPMNEdge id="BPMNEdge_SequenceFlow_12" bpmnElement="SequenceFlow_12"
sourceElement="BPMNShape_EventBasedGateway_1" targetElement="ParallelGateway_1567msb_di">
        <di:waypoint xsi:type="dc:Point" x="320.0" y="362.0"/>
        <di:waypoint xsi:type="dc:Point" x="352.0" y="362.0"/>
        <di:waypoint xsi:type="dc:Point" x="385.0" y="362.0"/>
        <bpmndi:BPMNLabel id="BPMNLabel_7"/>
      </bpmndi:BPMNEdge>
      <bpmndi:BPMNEdge id="BPMNEdge_SequenceFlow_13" bpmnElement="SequenceFlow_13"
sourceElement="BPMNShape_EventBasedGateway_1" targetElement="BPMNShape_Task_1">
        <di:waypoint xsi:type="dc:Point" x="295.0" y="387.0"/>
        <di:waypoint xsi:type="dc:Point" x="295.0" y="530.0"/>
        <di:waypoint xsi:type="dc:Point" x="442.0" y="530.0"/>
        <bpmndi:BPMNLabel id="BPMNLabel_17"/>
      </bpmndi:BPMNEdge>
      <bpmndi:BPMNEdge id="BPMNEdge_SequenceFlow_14" bpmnElement="SequenceFlow_14"
sourceElement="_BPMNShape_StartEvent_2" targetElement="BPMNShape_EventBasedGateway_1">
        <di:waypoint xsi:type="dc:Point" x="173.0" y="362.0"/>
        <di:waypoint xsi:type="dc:Point" x="221.0" y="362.0"/>
        <di:waypoint xsi:type="dc:Point" x="270.0" y="362.0"/>
        <bpmndi:BPMNLabel id="BPMNLabel_18"/>
      </bpmndi:BPMNEdge>
      <bpmndi:BPMNEdge id="BPMNEdge_DataOutputAssociation_7"
bpmnElement="DataOutputAssociation_6" sourceElement="BPMNShape_ReceiveTask_2"
targetElement="BPMNShape_DataObject_5">
        <di:waypoint xsi:type="dc:Point" x="552.0" y="444.0"/>
        <di:waypoint xsi:type="dc:Point" x="575.0" y="444.0"/>
        <di:waypoint xsi:type="dc:Point" x="598.0" y="444.0"/>
        <bpmndi:BPMNLabel id="BPMNLabel_19"/>
      </bpmndi:BPMNEdge>
      <bpmndi:BPMNEdge id="BPMNEdge_DataOutputAssociation_15"
bpmnElement="DataOutputAssociation_23" sourceElement="_BPMNShape_StartEvent_2"
targetElement="BPMNShape_DataStoreReference_3">
        <di:waypoint xsi:type="dc:Point" x="155.0" y="380.0"/>
        <di:waypoint xsi:type="dc:Point" x="155.0" y="436.0"/>
        <di:waypoint xsi:type="dc:Point" x="230.0" y="436.0"/>
        <bpmndi:BPMNLabel id="BPMNLabel_30"/>
      </bpmndi:BPMNEdge>
      <bpmndi:BPMNEdge id="BPMNEdge_DataOutputAssociation_16"
bpmnElement="DataOutputAssociation_24" sourceElement="_BPMNShape_StartEvent_2"
targetElement="BPMNShape_DataStoreReference_5">
        <di:waypoint xsi:type="dc:Point" x="155.0" y="344.0"/>
        <di:waypoint xsi:type="dc:Point" x="155.0" y="205.0"/>
        <di:waypoint xsi:type="dc:Point" x="230.0" y="205.0"/>
        <bpmndi:BPMNLabel id="BPMNLabel_54"/>
      </bpmndi:BPMNEdge>
      <bpmndi:BPMNEdge id="BPMNEdge_DataInputAssociation_6"
bpmnElement="DataInputAssociation_15" sourceElement="BPMNShape_DataStoreReference_5"
targetElement="BPMNShape_ReceiveTask_1">
        <di:waypoint xsi:type="dc:Point" x="280.0" y="205.0"/>
        <di:waypoint xsi:type="dc:Point" x="478.0" y="205.0"/>
        <di:waypoint xsi:type="dc:Point" x="478.0" y="269.0"/>
        <bpmndi:BPMNLabel id="BPMNLabel_55"/>
      </bpmndi:BPMNEdge>
      <bpmndi:BPMNEdge id="BPMNEdge_DataInputAssociation_7"
bpmnElement="DataInputAssociation_16" sourceElement="BPMNShape_DataStoreReference_4"
targetElement="BPMNShape_ReceiveTask_1">
        <di:waypoint xsi:type="dc:Point" x="280.0" y="286.0"/>
        <di:waypoint xsi:type="dc:Point" x="361.0" y="286.0"/>
        <di:waypoint xsi:type="dc:Point" x="442.0" y="285.0"/>
        <bpmndi:BPMNLabel id="BPMNLabel_56"/>
      </bpmndi:BPMNEdge>
      <bpmndi:BPMNEdge id="BPMNEdge_DataInputAssociation_8"
bpmnElement="DataInputAssociation_17" sourceElement="BPMNShape_DataStoreReference_3"
targetElement="BPMNShape_ReceiveTask_2">
        <di:waypoint xsi:type="dc:Point" x="280.0" y="436.0"/>
        <di:waypoint xsi:type="dc:Point" x="326.0" y="436.0"/>
        <di:waypoint xsi:type="dc:Point" x="384.0" y="436.0"/>
        <di:waypoint xsi:type="dc:Point" x="442.0" y="436.0"/>
        <bpmndi:BPMNLabel id="BPMNLabel_57"/>
      </bpmndi:BPMNEdge>
    </bpmndi:BPMNPlane>
    <bpmndi:BPMNLabelStyle id="BPMNLabelStyle_1">
      <dc:Font name="arial" size="9.0"/>
    </bpmndi:BPMNLabelStyle>
  </bpmndi:BPMNDiagram>
</bpmn:definitions>
```

Appendix C: Generated Smart Contract for the Loan Assessment Example using Caterpillar and Proposed Approach

C1. Generated Smart Contract for Loan Assessment Example Using Caterpillar

BPM17_Running_ExampleData contract

```
pragma solidity ^0.4.25;

import 'IData';

contract BPM17_Running_ExampleData is IDataImp {
    bool applicantEligible;
    uint monthlyRevenue;
    uint loanAmount;
    uint cost;
    uint appraisePropertyResult;
    uint assessLoanRiskResult;

    function executeScript(uint eInd) public returns(uint) {
        if (eInd == 4) {
            if (assessLoanRiskResult >= appraisePropertyResult)
                applicantEligible = true;
            else
                applicantEligible = false;
            return 8;
        }
        if (eInd == 5) {
            if (applicantEligible)
                return 16;
        }
        if (eInd == 12) {
            if (applicantEligible)
                return 4096;
        }
    }

    function checkIn(uint eInd, uint i1, uint i2, uint i3) public {
        require (4 & (1 << eInd) != 0);
        monthlyRevenue = i3; loanAmount = i2; cost = i1;
        continueExecution(eInd);
    }

    function checkIn(uint eInd, uint i1) public {
        require (1032 & (1 << eInd) != 0);

        if (eInd == 3) {
            assessLoanRiskResult = i1;
        }
        else if (eInd == 10) {
            appraisePropertyResult = i1;
        }
        continueExecution(eInd);
    }

    function checkIn(uint eInd, bool i1) public {
        require (2048 & (1 << eInd) != 0);
        applicantEligible = i1;
        continueExecution(eInd);
    }

    function checkOut(uint eInd) public view  {
        require (3084 & (1 << eInd) == 0);
    }
```

```
}
ProcessRegistry Contract
pragma solidity ^0.4.25;

contract IFunct {
    // WorkList functions
    function updateRuntimeRegistry(address _runtimeRegistry) public;
    // Factory Functions
    function setWorklist(address _worklist) public;
    function startInstanceExecution(address processAddress) public;
    function newInstance(address parent, address globalFactory) public
returns(address);
    function findParent() public view returns(address);
}

contract ProcessRegistry {

    mapping (bytes32 => mapping (uint => bytes32)) private
parent2ChildrenBundleId;
    mapping (bytes32 => address) private factories;
    mapping (bytes32 => bytes32) private policy;
    mapping (bytes32 => bytes32) private taskRole;

    mapping (address => bytes32) private instance2Bundle;
    mapping (address => address) private instance2PolicyOp;
    address[] private instances;

    mapping (address => bytes32) private worklist2Bundle;

    event NewInstanceCreatedFor(address parent, address processAddress);

    function registerFactory(bytes32 bundleId, address factory) external {
        factories[bundleId] = factory;
    }

    function registerWorklist(bytes32 bundleId, address worklist) external {
        address factory = factories[bundleId];
        require(factory != address(0));
        worklist2Bundle[worklist] = bundleId;
        IFunct(factory).setWorklist(worklist);
        IFunct(worklist).updateRuntimeRegistry(this);
    }

    function findRuntimePolicy(address pCase) public view returns(address) {
        return instance2PolicyOp[pCase];
    }

    function relateProcessToPolicy(bytes32 bundleId, bytes32 _policy, bytes32
_taskRole) external {
        taskRole[bundleId] = _taskRole;
        policy[bundleId] = _policy;
    }


    function addChildBundleId(bytes32 parentBundleId, bytes32 processBundleId,
uint nodeIndex) external {
        parent2ChildrenBundleId[parentBundleId][nodeIndex] = processBundleId;
    }

    function newInstanceFor(uint nodeIndex, address parent) public
returns(address) {
        return
newBundleInstanceFor(parent2ChildrenBundleId[instance2Bundle[parent]][nodeIndex],
parent, instance2PolicyOp[parent]);
    }
    function newBundleInstanceFor(bytes32 bundleId, address parent, address
policyOpAddr) public returns(address) {
        address factory = factories[bundleId];
        require(factory != address(0));
        address processAddress = IFunct(factory).newInstance(parent, this);
        instance2Bundle[processAddress] = bundleId;
        instance2PolicyOp[processAddress] = policyOpAddr;
        instances.push(processAddress);
        IFunct(factory).startInstanceExecution(processAddress);
        emit NewInstanceCreatedFor(parent, processAddress);
        return processAddress;
    }

    function allInstances() external view returns(address[]) {
        return instances;
    }

    function bindingPolicyFor(address procInstance) external view
returns(bytes32) {
        bytes32 pId = instance2Bundle[procInstance];
        address pAddr = procInstance;
        while(policy[pId].length != 0) {
            pAddr = IFunct(pAddr).findParent();
            if(pAddr == 0)
                break;
            pId = instance2Bundle[pAddr];
        }
        return policy[pId];
    }

    function taskRoleMapFor(address procInstance) external view returns(bytes32)
{
        bytes32 pId = instance2Bundle[procInstance];
        address pAddr = procInstance;
        while(taskRole[pId].length != 0) {
            pAddr = IFunct(pAddr).findParent();
            if(pAddr == 0)
                break;
            pId = instance2Bundle[pAddr];
        }
        return taskRole[pId];
    }

    function bindingPolicyFromId(bytes32 procId) external view returns(bytes32) {
        return policy[procId];
    }

    function taskRoleMapFromId(bytes32 procId) external view returns(bytes32) {
        return taskRole[procId];
    }

    function bundleFor(address processInstance) external view returns(bytes32) {
        return instance2Bundle[processInstance];
    }

    function childrenFor(bytes32 parent, uint nodeInd) external view
returns(bytes32) {
        return parent2ChildrenBundleId[parent][nodeInd];
    }

    function worklistBundleFor(address worklist) external view returns(bytes32) {
        return worklist2Bundle[worklist];
    }
}
```
BPMNInterpreter Contract

```
pragma solidity ^0.4.25;

import 'IData';
import 'IFlow';
import 'IFactory';

contract BPMNInterpreter {

    event MessageSent(bytes32 messageBody);
    event NewCaseCreated(address pCase);

    /// Instantiation of Root-Process
    function createInstance(address cFlow) public returns(address) {
        address factory = IFlow(cFlow).getFactoryInst();
        require(factory != address(0));

        address pCase = IFactory(factory).newInstance();
        IData(pCase).setParent(address(0), cFlow, 0);

        emit NewCaseCreated(pCase);

        executionRequired(cFlow, pCase);

    }

    /// Instantiation of a sub-process by its parent
    function createInstance(uint eInd, address pCase) private returns(address) {
        address pFlow = IData(pCase).getCFlowInst();
        address cFlow = IFlow(pFlow).getSubProcInst(eInd);

        address factory = IFlow(cFlow).getFactoryInst();
        require(factory != address(0));

        address cCase = IFactory(factory).newInstance();

        IData(cCase).setParent(pCase, cFlow, eInd);
        IData(pCase).addChild(eInd, cCase);

        executionRequired(cFlow, cCase);

        return cCase;
    }

    function executionRequired(address cFlow, address pCase) private {
        uint eFirst = IFlow(cFlow).getFirstElement();
        IData(pCase).setMarking(IFlow(cFlow).getPostCond(eFirst));

        uint[] memory next = IFlow(cFlow).getAdyElements(eFirst);

        if(next.length > 0)
            executeElements(pCase, next[0]);
    }

    function throwEvent(address pCase, bytes32 evCode, uint evInfo) private {

        /// This function only receive THROW EVENTS (throw event verification
made in function executeElement)

        uint[2] memory pState;
        pState[0] = IData(pCase).getMarking();
        pState[1] = IData(pCase).getStartedActivities();

        if(evInfo & 4096 == 4096)
        /// Message (BIT 15), to publish a Message in the Ethereum Event Log
            emit MessageSent(evCode);
        if(evInfo & 5632 == 5632) {
            /// 9- End, 10- Default, 12- Message
            if(pState[0] | pState[1] == 0) // If there are not tokens to consume
nor started activities in any subprocess
                tryCatchEvent(pCase, evCode, evInfo, true); // Sub-process ended,
thus continue execution on parent
        } else {
            if(evInfo & 2048 == 2048)
            /// Terminate Event (BIT 11), only END EVENT from standard,
                killProcess(pCase);  // Terminate the execution in the current
Sub-process and each children
            tryCatchEvent(pCase, evCode, evInfo, pState[0] | pState[1] == 0); //
Continue the execution on parent
        }
    }


    function tryCatchEvent(address pCase, bytes32 evCode, uint evInfo, bool
isInstCompl) private {

        address catchCase = IData(pCase).getParent();
        if(catchCase == address(0)) {
            /// No Parent exist, root node
            if(evInfo & 8192 == 8192)
            /// Error event (BIT 13), only END EVENT from standard, in the root
process.
                killProcess(pCase);
            return;
        }

        address cFlow = IData(catchCase).getCFlowInst();

        uint[2] memory pState;
        pState[0] = IData(catchCase).getMarking();
        pState[1] = IData(catchCase).getStartedActivities();

        uint subProcInd = IData(pCase).getIndexInParent();

        uint runInstCount = isInstCompl ?
IData(catchCase).decreaseInstanceCount(subProcInd)
        : IData(catchCase).getInstanceCount(subProcInd);

        if(runInstCount == 0)
        /// Update the corresponding sub-process, call activity as completed
            IData(catchCase).setActivityMarking(pState[1] & ~(1 << 1 <<
subProcInd));

        uint subProcInfo = IFlow(cFlow).getTypeInfo(subProcInd);

        if(evInfo & 7168 != 0) {
            /// If receiving 10- Default, 11- Terminate or 12- Message
            if(runInstCount == 0 && subProcInfo & 4096 != 4096) {
                /// No Instances of the sub-process propagating the event and The
sub-process isn't an event-sub-process (BIT 12)
                IData(catchCase).setMarking(pState[0] &
~IFlow(cFlow).getPostCond(subProcInd));
                executeElements(catchCase,
IFlow(cFlow).getAdyElements(subProcInfo)[0]);
            } else if(subProcInfo & 128 == 128) {
                /// Multi-Instance Sequential (BIT 7), with pending instances to
be started.
                createInstance(subProcInd, pCase);
            }
        } else {
            /// Signal, Error or Escalation

            /// Signals are only handled from the Root-Process by Broadcast, thus
```

```
the propagation must reach the Root-Process.
            if(evInfo & 32768 == 32768) {
                /// Propagating the Signal to the Root-Process
                while(catchCase != address(0)) {
                    pCase = catchCase;
                    catchCase = IData(pCase).getParent();
                }
                broadcastSignal(pCase);
                return;
            }

            uint[] memory events = IFlow(cFlow).getEventList();

            /// The event can be catched only once, unless it is a signal where a
broadcast must happen.
            /// Precondition: Event-subprocess must appear before boundary events
on the event list.
            for(uint i = 0; i < events.length; i++) {

                if(IFlow(cFlow).getEventCode(events[i]) == evCode) {
                    /// Verifiying there is a match with the throw-cath events.

                    uint catchEvInfo = IFlow(cFlow).getTypeInfo(events[i]);  //
Info of the candidate catching event
                    uint attachedTo = IFlow(cFlow).getAttachedTo(events[i]); //

                    if(catchEvInfo & 6 == 6) {
                        /// Start event-sub-process (BIT 6)
                        if(catchEvInfo & 16 == 16)
                        /// Interrupting (BIT 4 must be 1, 0 if non-interrupting)
                            killProcess(catchCase);  // Before starting the event
subprocess, the parent is killed

                        createInstance(attachedTo, pCase);  // Starting event
sub-process
                        IData(catchCase).setActivityMarking(pState[1] | (1 <<
attachedTo));  // Marking the event-sub-process as started
                        return;
                    } else if(catchEvInfo & 256 == 256 && attachedTo ==
subProcInd) {
                        /// Boundary (BIT 6) of the subproces propagating the
event
                        if(catchEvInfo & 16 == 16)
                        /// Interrupting (BIT 4 must be 1, 0 if non-interrupting)
                            killProcess(pCase);  // The subprocess propagating
the event must be interrupted

                        IData(catchCase).setMarking(pState[0] &
~IFlow(cFlow).getPostCond(events[i])); // Update the marking with the output of
the boundary event
                        executeElements(catchCase,
IFlow(cFlow).getAdyElements(events[i])[0]); // Continue the execution of possible
internal elements
                        return;
                    }
                }
            }
            /// If the event was not caught the propagation continues to the
parent unless it's the root process
            throwEvent(catchCase, evCode, evInfo);
        }
    }

    function killProcess(address pCase) private {
        uint startedActivities = IData(pCase).getStartedActivities();
        IData(pCase).setMarking(0);
        IData(pCase).setActivityMarking(0);

        uint[] memory children =
IFlow(IData(pCase).getCFlowInst()).getSubProcList();

        for(uint i = 0; i < children.length; i++)
            if(startedActivities & (1 << children[i]) != 0)
                killProcess(IData(pCase).getChildProcInst(children[i]));
    }

    function killProcess(address[] memory pCases) private {
        for(uint i = 0; i < pCases.length; i++)
            killProcess(pCases[i]);
    }


    function broadcastSignal(address pCase) private {

        address cFlow = IData(pCase).getCFlowInst();

        uint[] memory events = IFlow(cFlow).getEventList();

        uint[2] memory pState;
        pState[1] = IData(pCase).getStartedActivities();

        for(uint i = 0; i < events.length; i++) {
            uint evInfo = IFlow(cFlow).getTypeInfo(events[i]);

            if(evInfo & 32780 == 32772) {
                /// Event Catch Signal (BITs 2, 3 [0-catch, 1-throw], 15)

                uint catchEvInfo = IFlow(cFlow).getTypeInfo(events[i]);
                uint attachedTo = IFlow(cFlow).getAttachedTo(events[i]);

                if(catchEvInfo & 6 == 6) {
                    /// Start event-sub-process (BIT 6)
                    if(catchEvInfo & 16 == 16)
                    /// Interrupting (BIT 4 must be 1, 0 if non-interrupting)
                        killProcess(pCase);  // Before starting the event
subprocess, the current process-instance is killed

                    createInstance(attachedTo, pCase);  // Starting event sub-
process
                    IData(pCase).setActivityMarking(1 << attachedTo);  // Marking
the event-sub-process as started
                } else if(catchEvInfo & 256 == 256) {
                    /// Boundary (BIT 6) of the subproces propagating the event
                    if(catchEvInfo & 16 == 16)
                    /// Interrupting (BIT 4 must be 1, 0 if non-interrupting)
                        killProcess(IData(pCase).getChildProcInst(attachedTo));
// The subprocess propagating the event must be interrupted

                    IData(pCase).setMarking(IData(pCase).getMarking() &
~IFlow(cFlow).getPostCond(events[i])); // Update the marking with the output of
the boundary event
                    executeElements(pCase,
IFlow(cFlow).getAdyElements(events[i])[0]); // Continue the execution of possible
internal elements
                } else if(evInfo & 160 == 160) {
                    /// Start (not Event Subprocess) OR Intermediate Event
                    IData(pCase).setMarking(IData(pCase).getMarking() &
~IFlow(cFlow).getPreCond(events[i]) | IFlow(cFlow).getPostCond(events[i]));
                    executeElements(pCase,
IFlow(cFlow).getAdyElements(events[i])[0]);
                }
            }
        }

        uint[] memory children =
IFlow(IData(pCase).getCFlowInst()).getSubProcList();
        uint startedActivities = IData(pCase).getStartedActivities();

        for(uint j = 0; j < children.length; j++)
            if(startedActivities & (1 << children[j]) != 0)
                broadcastSignal(IData(pCase).getChildProcInst(children[j]));
    }

    function broadcastSignal(address[] memory pCases) private {
        for(uint i = 0; i < pCases.length; i++)
            broadcastSignal(pCases[i]);
    }


    function executeElements(address pCase, uint eInd) public {
        address cFlow = IData(pCase).getCFlowInst();
        uint[] memory next;

        /// Declared as an array and not as independent fields to avoid Stack Too
Deep- Compilation Error

        /// 0- preC
        /// 1- postC
        /// 2- typeInfo
        uint[3] memory lInfo;

        /// 0- tokensOnEdges
        /// 1- startedActivities
        uint[2] memory pState;

        pState[0] = IData(pCase).getMarking();
        pState[1] = IData(pCase).getStartedActivities();

        /// Execution queue and pointers to the first & last element (i.e. basic
circular queue implementation)
        uint[100] memory queue;
        uint i = 0; uint count = 0;

        queue[count++] = eInd;

        while (i < count) {

            eInd = queue[i++];

            (lInfo[0], lInfo[1], lInfo[2], next) =
IFlow(cFlow).getElementInfo(eInd);
            // (preC, postC, typeInfo, next) = IFlow(cFlow).getElementInfo(eInd);

            /// Verifying Preconditions (i.e. Is the element enabled?)


            if(lInfo[2] & 42 == 42 ) {
                /// else if (AND Join)
                if(pState[0] & lInfo[0] != lInfo[0])
                    continue;
                pState[0] &= ~lInfo[0];
            } else if(lInfo[2] & 74 == 74 ) {
                /// else if (OR Join)
                ///// OR Join Implementation //////
            } else if(lInfo[2] & 1 == 1 || (lInfo[2] & 4 == 4 && lInfo[2] & 640
!= 0)  || lInfo[2] & 2 == 2) {
                /// If (Activity || Intermediate/End Event || Gateway != AND/OR
Join)
```

```
                if(pState[0] & lInfo[0] == 0)
                    continue;
                pState[0] &= ~lInfo[0];   // Removing tokens from input arcs
            } else {
                continue;
            }

            /// Executing current element (If enabled)

            if(lInfo[2] & 65 == 65) {
                /// (0- Activity, 6- Parallel Multi-Instance)
                uint cInst = IFlow(cFlow).getInstanceCount(eInd);
                while(cInst-- > 0)
                    createInstance(eInd, pCase);
                pState[1] |= (1 << eInd);
            } else if(lInfo[2] & 129 == 129 || (lInfo[2] & 1 == 1 && lInfo[2] &
48 != 0 && lInfo[2] & 4096 == 0)) {
                /// If (0- Activity, 7- Sequential Multi-Instance) ||
                /// Sub-process(0- Activity, 5- Sub-process) or Call-Activity(0-
Activity, 4- Call-Activity)
                /// but NOT Event Sub-process(12- Event Subprocess)
                IData(createInstance(eInd, pCase)).setInstanceCount(eInd,
IFlow(cFlow).getInstanceCount(eInd));
                pState[1] |= (1 << eInd);
            } else if(lInfo[2] & 4105 == 4105 || (lInfo[2] & 10 == 2 && lInfo[2]
& 80 != 0)) {
                /// (0- Activity, 3- Task, 12- Script) ||
                /// Exclusive(XOR) Split (1- Gateway, 3- Split(0), 4- Exclusive)
||
                /// Inclusive(OR) Split (1- Gateway, 3- Split(0), 6- Inclusive)
                IData(pCase).executeScript(eInd);
                pState[0] |= IData(pCase).executeScript(eInd);
            } else if ((lInfo[2] & 9 == 9 && lInfo[2] & 27657 != 0) || lInfo[2] &
2 == 2) {
                /// If (User(11), Service(13), Receive(14) or Default(10) Task ||
Gateways(1) not XOR/OR Split)
                /// The execution of User/Service/Receive is triggered off-chain,
                /// Thus the starting point would be the data contract which
executes any script/data-update related to the task.
                pState[0] |=  lInfo[1];
            } else if(lInfo[2] & 12 == 12) {
                /// If (2- Event, 3- Throw(1))
                IData(pCase).setMarking(pState[0]);
                IData(pCase).setActivityMarking(pState[1]);
                throwEvent(pCase, IFlow(cFlow).getEventCode(eInd), lInfo[2]);
                if(IData(pCase).getMarking() |
IData(pCase).getStartedActivities() == 0)
                /// By throwing the event, a kill was performed so the current
instance was terminated
                    return;
                pState[0] = IData(pCase).getMarking();
                pState[1] = IData(pCase).getStartedActivities();

                if(lInfo[2] & 128 == 128)
                /// If Intermediate event (BIT 7)
                    pState[0] |=  lInfo[1];
            }

            /// Adding the possible candidates to be executed to the queue.
            /// The enablement of the element is checked at the moment it gets
out of the queue.
            for(uint j = 0; j < next.length; j++) {
                queue[count] = next[j];
                count = ++count % 100;
            }
        }
        /// Updating the state (storage) after the execution of each internal
element.
        IData(pCase).setMarking(pState[0]);
        IData(pCase).setActivityMarking(pState[1]);

    }
}
```

IFlow and IFlowImpl contracts

```
pragma solidity ^0.4.25;

contract IFlow {

    function getPreCond(uint eInd) public view returns(uint);
    function getPostCond(uint eInd) public view returns(uint);
    function getTypeInfo(uint eInd) public view returns(uint);
    function getAdyElements(uint eInd) public view returns(uint[] memory);
    function getEventCode(uint eInd) public view returns(bytes32);
    function getAttachedTo(uint eInd) public view returns(uint);

    function getElementInfo(uint eInd) public view returns(uint, uint, uint,
uint[] memory);
    function getFirstElement() public view returns(uint);

    function getEventList() public view returns(uint[] memory);
    function getSubProcList() public view returns(uint[] memory);
    function getSubProcInst(uint eInd) public view returns(address);
    function getFactoryInst() public view returns(address);
    function setFactoryInst(address _factory) public;
    function getInstanceCount(uint eInd) public view returns(uint);

    function getInterpreterInst() public view returns(address);
    function setInterpreterInst(address _interpreter) public;

    function setElement(uint eInd, uint preC, uint postC, uint typeInfo, bytes32
eCode, uint[] memory _nextElem) public;
    function linkSubProcess(uint pInd, address cFlowInst, uint[] memory
attachedEvt, uint countInstances) public;

}

/// @title Control Flow Data Structure
/// @dev Tree mirroring the process hierarchy derived from a process model.
/// @dev This contract must be deployed/instantiated once per process model.
contract IFlowImpl {

    uint private startEvt;
    address private factory;
    address private interpreter;

    // elemIndex => [preC, postC, type]
    mapping(uint => uint[3]) private condTable;

    // Element Index => List of elements that can be enabled with the completion
of the key element
    mapping(uint => uint[]) private nextElem;

    // List of Indexes of the subprocesses
    uint[] private subProcesses;

    // List of Event Indexes defined in the current Subprocess
    uint[] private events;

    // Event Index => Index of the element where event is attachedTo
    mapping(uint => uint) private attachedTo;
    // Event Index => String representing the code to identify the event (for
catching)
    mapping(uint => bytes32) private eventCode;

    // Subprocess Index => Child Subproces address
    mapping(uint => address) private pReferences;

    // Subprocess Index => number of instances
    mapping(uint => uint) private instanceCount;


    function getPreCond(uint eInd) public view returns(uint) {
        return condTable[eInd][0];
    }

    function getPostCond(uint eInd) public view returns(uint) {
        return condTable[eInd][1];
    }

    function getTypeInfo(uint eInd) public view returns(uint) {
        return condTable[eInd][2];
    }

    function getFirstElement() public view returns(uint) {
        return startEvt;
    }

    function getAdyElements(uint eInd) public view returns(uint[] memory) {
        return nextElem[eInd];
    }

    function getElementInfo(uint eInd) public view returns(uint, uint, uint,
uint[] memory) {
        return (condTable[eInd][0], condTable[eInd][1], condTable[eInd][2],
nextElem[eInd]);
    }

    function getSubProcList() public view returns(uint[] memory) {
        return subProcesses;
    }

    function getInstanceCount(uint eInd) public view returns(uint) {
        return instanceCount[eInd];
    }

    function getEventCode(uint eInd) public view returns(bytes32) {
        return eventCode[eInd];
    }

    function getEventList() public view returns(uint[] memory) {
        return events;
    }

    function getAttachedTo(uint eInd) public view returns(uint) {
        return attachedTo[eInd];
    }

    function getSubProcInst(uint eInd) public view returns(address) {
        return pReferences[eInd];
    }

    function getFactoryInst() public view returns(address) {
        return factory;
    }

    function setFactoryInst(address _factory) public {
        factory = _factory;
    }

    function getInterpreterInst() public view returns(address) {
        return interpreter;
    }

    function setInterpreterInst(address _interpreter) public {
        interpreter = _interpreter;
    }

    function setElement(uint eInd, uint preC, uint postC, uint typeInfo, bytes32
eCode, uint[] memory _nextElem) public {
        uint _typeInfo = condTable[eInd][2];
        if (_typeInfo == 0) {
            if (typeInfo & 4 == 4) {
                events.push(eInd);
                if (typeInfo & 36 == 36)
                    startEvt = eInd;
                eventCode[eInd] = eCode;
            } else if (typeInfo & 33 == 33)
                subProcesses.push(eInd);
        } else
            require (_typeInfo == typeInfo);
        condTable[eInd] = [preC, postC, typeInfo];
        nextElem[eInd] = _nextElem;

    }

    function linkSubProcess(uint pInd, address cFlowInst, uint[] memory
attachedEvt, uint countInstances) public {
        require(condTable[pInd][2] & 33 == 33);  // BITs (0, 5) Veryfing the
subprocess to link is already in the data structure
        pReferences[pInd] = cFlowInst;
        for(uint i = 0; i < attachedEvt.length; i++)
            if(condTable[attachedEvt[i]][2] & 4 == 4)
                attachedTo[attachedEvt[i]] = pInd;
        instanceCount[pInd] = countInstances;
    }
}
```

BPM17_Running_ExampleFactory contract

```
pragma solidity ^0.4.25;

import 'BPM17_Running_ExampleData';

contract BPM17_Running_ExampleFactory {

    function newInstance() public returns(address) {
        return new BPM17_Running_ExampleData();
    }

}
```

C2. Generated Smart Contract for Loan Assessment Example Using Proposed Approach

```
pragma solidity >=0.4.0 <0.7.0;

import "github.com/oraclize/ethereum-api/oraclizeAPI.sol";

contract LoanApproval is usingOraclize {
    mapping(bytes32 => uint) validIds;
    uint public cost;
    uint public loanAmount;
    uint public monthlyRevenue;
    uint loanRisk = 0;
    uint appraiseProperty = 0;

    event ConfirmationSent();
    event LoanRejected();
    event LoanAccepted();

    constructor (uint _cost, uint _loanAmount, uint _monthlyRevenue) {
        loanAmount = _loanAmount;
        cost = _cost;
        monthlyRevenue = _monthlyRevenue;
        emit LoanRejected();

        bytes32 _queryId = 1;
        //        bytes32 _queryId = oraclize_query("URL",
'json(https://assess.loan.risk.url).result');
        validIds[_queryId] = 1;
        //         _queryId = oraclize_query("URL",
'json(https://assess.loan.risk.url).result');
        validIds[_queryId] = 2;

        if (loanRisk >= appraiseProperty){
            emit ConfirmationSent();
        } else {
            emit LoanRejected();
        }


        if (loanRisk >= appraiseProperty){
            emit ConfirmationSent();
        } else {
            emit LoanRejected();
        }
    }

    function confirmAcceptance(bool _confirmation) {
        if (_confirmation == true){
            emit LoanAccepted();
        }
    }

    function __callback(bytes32 _queryId, string memory _result) public {
        require(msg.sender == oraclize_cbAddress());
        if (validIds[_queryId] == 1) {
            appraiseProperty = parseInt(_result);
            delete validIds[_queryId];
        }
        if (validIds[_queryId] == 2) {
            loanRisk = parseInt(_result);
            delete validIds[_queryId];
        }
    }

}
```

Appendix D: Truffle Tests Written for the Re-generated Solidity Code

D1: Truffle Test Written for Invoice Handling Smart Contract

```
const InvoiceHandling = artifacts.require('InvoiceHandling');

contract('InvoiceHandling', (accounts) => {
    let invoiceHandling;
    before(async () => {
        invoiceHandling = await InvoiceHandling.new({ from: accounts[0] });
    });

    it("should be able to issue invoice", async () => {
        await invoiceHandling.issueInvoice();
    });

    it("should be able to disapprove invoice", async () => {
        await invoiceHandling.approveInvoice(false);
    });

    it("should be able to fixInvoice invoice", async () => {
        await invoiceHandling.fixInvoice();
    });

    it("should be able to approve invoice", async () => {
        await invoiceHandling.approveInvoice(true);
    });
});
```

D2: Truffle Test Written for Lottery Smart Contract

```
const Lottery10Users = artifacts.require('Lottery10Users');
let catchRevert = require("./exceptions.js").catchRevert;
var BN = web3.utils.BN;

contract('Lottery10Users', (accounts) => {
    let lottery10Users;
    let users = [];
    before(async () => {
        lottery10Users = await Lottery10Users.new({ from: accounts[0] });
    });

    it("should revert join when no value provided", async () => {
        await catchRevert(lottery10Users.join({ from: accounts[0] }));
    });

    it("should accept join when 0.1 ether value provided", async () => {
        var amount = 0.1;
        await lottery10Users.join({ value:
web3.utils.toBN(web3.utils.toWei(amount.toString(), 'ether')), from: accounts[0]
});
    });

    it("should accept join more 9 members", async () => {
        var amount = 0.1;
        for (var i = 1; i < 10; i++) {
            await lottery10Users.join({ value:
web3.utils.toBN(web3.utils.toWei(amount.toString(), 'ether')), from: accounts[i]
});
        }
    });

    it("should one member win the jackpot", async () => {
        var foundWinner = false;
        for (var i = 0; i < 10; i++) {
            let balance = await web3.eth.getBalance(accounts[i]);
            if (balance > 100000000000000000000) {
                foundWinner = true;
            }
        }
        assert.equal(foundWinner, true);
    });
});
```

D3: Truffle Test Written for Marriage Smart Contract

```
const Marriage = artifacts.require('Marriage');

contract('Marriage', (accounts) => {
    let marriage;
    const contributor = accounts[1];
    const wifeAccount = accounts[2];
    const husbandAccount = accounts[3];

    // build up and tear down a new BasicToken contract before each test
    beforeEach(async () => {
        marriage = await Marriage.new(wifeAccount, husbandAccount, { from:
contributor });
    });

    it("should be equal the wifeAccount and the initial balance", async () => {
        assert.equal(100000000000000000000, await
web3.eth.getBalance(wifeAccount));
```

Appendix E: Detailed Responses of the Survey

Table 5.1: Detailed Responses for the Survey Study

Example	Question	Answers (%) Exc.	Good	Avg.	Poor	Very Poor	Free Text Feedback
Rental Agreement	1. Your Score for the **Readability** of the input Diagram	14.3	57.1	28.6	0	0	– Based on the knowledge – The language used for defining objects is understandable – Consists the logical expressions in illustration well – understood the logic – Quite clear and relationships are well illustrated – Isolated notations, unnamed relationships – Its similar to a flow diagram
	2. Your Score for the Correctness of the **Business Logic** of the BPMN-to-Solidity Translation	14.3	28.6	42.9	14.3	0	– Seems to cover all the scenarios in the diagram – I think you have used modifiers in Solidity to represent if-else branching in BPMN. However, it's somewhat difficult for me to understand. – The diagram is directly converted. However the flow is not represented in the code. – not much familiar – State transitions are correctly stated – Number of logic aligns with the diagram – couldnt figure out the construct
	3. Your Score for the Correctness of the **State Transitions** of the	28.6	28.6	0	42.8	0	– i hardly notice the state transition is implemented – I can identify the mapping of variables and methods between BPMN-to-Solidity to a greater extent – All objects are properly transmitted.

	BPMN-to-Solidity Translation						– Not using for the current job role – State transitions have been correctly modeled – Good translation - e.g. variable names – We required to know the formal definition in order to verify
	4. Your Score for the Correctness of the **Entry Points** of the BPMN-to-Solidity Translation	28.6	42.8	28.6	0	0	– Seems to cover all the scenarios in the diagram – Public methods are properly defined in BPMN as well as in Solidity. – Clearly marked in both BPMN and Solidity – Got the logic – Most of the entry points are well stated – correct translation – We required We required to know the formal definition in order to verify
Marriage: Wedding Gift	1. Your Score for the **Readability** of the input Diagram	14.3	57.1	14.3	14.3	0	– This is seems to be fuzzy to me – Difficult to figure out the exact scenario – Connection of the variables is not represented. The connection between Marriage and account balances is confusing as well. – Got the logic – Quiet clear and readable – what is pool1? – simple scenario
	2. Your Score for the Correctness of the **Business Logic** of the	14.3	28.6	42.8	14.3	0	– Seems to me i am lost hr – Difficult to identify the if-else portion in the diagram – The hidden details of BPMN is clearly expressed in the code.

	BPMN-to-Solidity Translation						– Not much familiar – Decision point can be improved – unable to determine - no if statements – since its a simple scenario, its easy to verify
	3. Your Score for the Correctness of the **State Transitions** of the BPMN-to-Solidity Translation	0	42.8	28.6	28.6	0	– Seems to me i am lost hr – No clear enough representations – Balance state transitions is clearly mentioned – Not much familiar – State transitions are well illustrated – good translation – since its a simple scenario, its easy to verify
	4. Your Score for the Correctness of the **Entry Points** of the BPMN-to-Solidity Translation	14.3	28.6	42.8	14.3	0	– Seems to me i am lost hr – Can be identified to a some extent, but I feel the diagram should be more detailed – Function wise decoupling has achieved this. – not much familiar – Could improve the representations of the entry points – all covered – since its a simple scenario, its easy to verify
Sellable	1. Your Score for the **Readability** of the input Diagram	42.8	14.3	42.8	0	0	– Input is true to the scenario – Objects are properly defined and can easily be understood – Not Clear – Got the logic – Quiet clear and meaningful – readable - no tangling items

							– Complex scenario
	2. Your Score for the Correctness of the **Business Logic** of the BPMN-to-Solidity Translation	14.3	28.6	28.6	28.6	0	– cover all the scenarios in the diagram – I couldn't properly find if-else tags – The BPMN flows are correctly visible – not much familiar – Business logic are correctly captured and modeled – unable to determine – Complex scenario
	3. Your Score for the Correctness of the **State Transitions** of the BPMN-to-Solidity Translation	28.6	28.6	28.6	14.3	0	– state transition is covers as to the scenario – They can properly be identified within the diagram and mapped Solidity code – Catches the states of the respective objects – not much familiar – Most of the state transitions are captured, yet can improve – good – Complex scenario
	4. Your Score for the Correctness of the **Entry Points** of the BPMN-to-Solidity Translation	28.6	42.8	14.3	14.3	0	– cover all the scenarios in the diagram – Can easily be identified – not much familiar – Almost all of the entry points are clearly identified – good – Complex scenario
Lotte ry	1. Your Score for the **Readability** of the input Diagram	14.3	28.6	57.1	0	0	– covers all the scenarios – Given business logic is properly mapped into the diagram

							– All representations are better – got the logic – Generated code is quiet clear – what is pool 1 – Complex scenario
	2. Your Score for the Correctness of the **Business Logic** of the BPMN-to-Solidity Translation	28.6	28.6	28.6	14.3	0	– covers all the scenarios to best of my knowledge – I think the mapping almost covered the given scenarios – Translation is correct and readable – not much familiar – All the business logic are captured – all covered – Complex scenario
	3. Your Score for the Correctness of the **State Transitions** of the BPMN-to-Solidity Translation	0	57.1	28.6	14.3	0	– program dose't depend much on state transition – The data objects in storing variables can't be seen in this diagram as opposed to the previous diagrams – All elements are captured – not much familiar – State transitions of the objects are correctly identified – mostly covered – Complex scenario
	4. Your Score for the Correctness of the **Entry Points** of the BPMN-to-Solidity Translation	57.1	14.3	14.3	14.3	0	– it has all the entry points that are in the diagram – Those are properly defined; therefore, can easily be identified – All elements are captured – not much familiar – All the entry points are correctly captured – good job

							– Complex scenario
Basic Token	1. Your Score for the **Readability** of the input Diagram	28.6	14.3	42.8	14.3	0	– covers all the scenarios – It's hard for me to identify the scenario from the given diagram – All cases captured – Got the idea – All the points are clearly identified and included in the code – isolated items – Complex scenario
	2. Your Score for the Correctness of the **Business Logic** of the BPMN-to-Solidity Translation	14.3	28.6	28.6	28.6	0	– covers all the scenarios to best of my knowledge – Same as above – Good coversion. Small gaps in readability – not much familiar – Business logic are clearly depicted – unable to determine – Complex scenario
	3. Your Score for the Correctness of the **State Transitions** of the BPMN-to-Solidity Translation	14.3	42.8	14.3	28.6	0	– covers all the scenarios to best of my knowledge – Same as above – Translation is to the point – not much familiar – All the state transitions are well captured and demonstrated – all covered – Complex scenario

	4. Your Score for the Correctness of the **Entry Points** of the BPMN-to-Solidity Translation	28.6	28.6	14.3	28.6	0	– covers all the scenarios to best of my knowledge – Same as above – All entry points are shown correctly – not much familiar – All the entry points are captured – good – Complex scenario
Crowd Sale	1. Your Score for the **Readability** of the input Diagram	28.6	14.3	42.8	14.3	0	– Covers the explained scenario – My score would be 50% for this diagram and I feel we need to have the detailed business logic to make a more accurate comment – Complex usecase nicely illustrated in diagram – Got the idea – Well structured and clear – pool 1, tangling items, isolated items – Complex scenario
	2. Your Score for the Correctness of the **Business Logic** of the BPMN-to-Solidity Translation	14.3	28.6	28.6	28.6	0	– correct based on my knowledge – Same as above – Captures all the cases – not much familiar – All the business logic are captured – good – Complex scenario
	3. Your Score for the Correctness of **the State Transitions** of the	14.3	28.6	42.9	14.3	0	– state transition is not playing major part in the diagram or in the code in this scenario – Same as above – Transitions are capture well

	BPMN-to-Solidity Translation						– not much familiar – State transitions are well identified – good – Complex scenario
	4. Your Score for the Correctness of the **Entry Points** of the BPMN-to-Solidity Translation	28.6	28.6	28.6	28.6	0	– correct based on my knowledge – Same as above – All entry points are represented in the translation – not much familiar – Most of the entry points are captured – all covered – Complex scenario