CEUR-WS.org/Vol-2196/BPM_2018_paper_12.pdf

Lorikeet: A Model-Driven Engineering Tool for Blockchain-Based Business Process Execution and Asset Management

An Binh Tran[1], Qinghua Lu[1,2], and Ingo Weber[1,2]

[1] Data61, CSIRO, Sydney, Australia
[2] School of Computer Science and Engineering, UNSW, Sydney, Australia
`firstname.lastname@data61.csiro.au`

Abstract. Blockchain has attracted broad interest as a distributed ledger technology for building the next generation of applications to address lack-of-trust issues in business. Business processes that manage assets (e.g. transferring car/grain/land titles) are a promising domain for applying blockchain technology: secure asset management (including tokens and crypto-currency) is a major application area of blockchain. Solutions for non-fungible assets (like cars and houses, but also supply chain goods) are recurrently addressed case by case and traditionally rely on a centralised trusted authority. This can cause trust issues and introduce inefficiencies or counterparty risks. However, developing blockchain applications is far from easy, and mistakes may not be fixable. Thus, in this paper, we present a model-driven engineering (MDE) tool named *Lorikeet* for the implementation of business processes on blockchain, which can manage assets. Lorikeet can automatically create well-tested smart contract code from specifications that are encoded in the business process and data registry models based on the implemented model transformations. We demonstrate the tool with an industrial use case.

Keywords: Blockchain · Model-Driven Engineering · Business Process · Registry

1 Introduction

Blockchain is an innovative distributed ledger technology that enables agreements on decentralised and transactional data sharing across a large network of untrusted participants without relying on a central trusted authority. Many organisations (e.g. startups, enterprises, and governments) [1, 6] are currently exploring how to leverage blockchain technology to build trust in the next generation of applications. Blockchain application areas are diverse, including physical or digital asset registry, tokens, currency, identity management, and business processes.

Management of assets is considered to be the first "killer application" of blockchain, starting with fungible assets like crypto-currency (Bitcoin, Ether,

F. Casati et al. (Eds.): Proceedings of the Dissertation Award and Demonstration, Industrial Track at BPM 2018, CEUR-WS.org, 2018. Copyright © 2018 for this paper by its authors. Copying permitted for private and academic purposes. This volume is published and copyrighted by its editors.

etc.) and tokens (second-tier coins that are managed on existing blockchain networks like Ethereum). Non-fungible assets (e.g., car or house titles, and titles to supply chain goods) can also be managed using smart contract technology. In contrast to fungible assets, the latter are recurrently solved case by case and are traditionally managed by relying on a centralised trusted authority. This can cause trust issues and introduce inefficiencies or counterparty risks (e.g., re-assigning ownership of goods before payment).

However, building systems on blockchain is non-trivial due to the steep learning curve of the blockchain technology. According to a survey by Gartner [4], "23 percent of [relevant surveyed] CIOs said that blockchain requires the most new skills to implement of any technology area, while 18 percent said that blockchain skills are the most difficult to find." Model-driven engineering (MDE) is software engineering methodology that automatically creates software system code from the models. MDE tools can generate well-tested code implementing best practices and help developers manage software complexity by only focusing on building high-level models without requiring expert development knowledge [5].

In this paper, we present how to leverage MDE to facilitate the development of blockchain applications in the space of business processes and asset registries, which can easily be applied to a broad range of blockchain applications. We designed and implement an MDE tool, named Lorikeet, which can automatically produce smart contracts from business process models and registry data schema. Lorikeet incorporates the registry editor Regerator [7] and implements the BPMN translation algorithms from both [8] and [2]. It is an entirely separate tool from Caterpillar [3], and tackles the hard challenges of integrating the asset management and business process interactions on blockchain. Lorikeet is in commercial use by Data61, and has been applied in numerous industry projects.

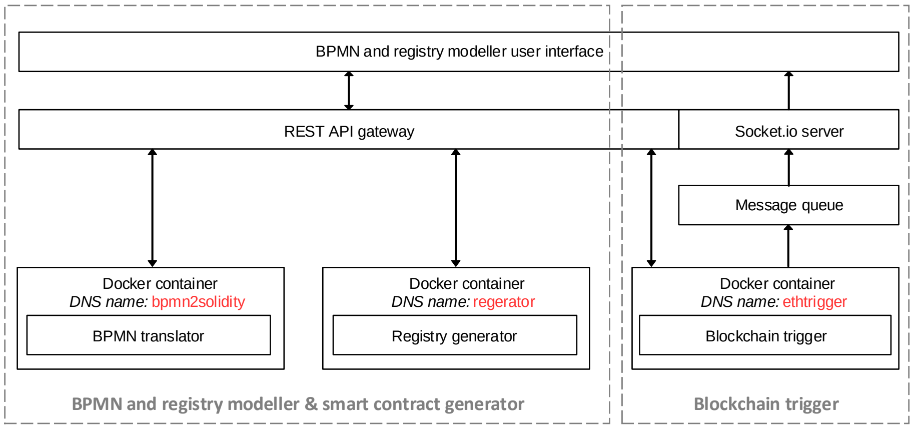


Fig. 1. Architecture of Lorikeet.

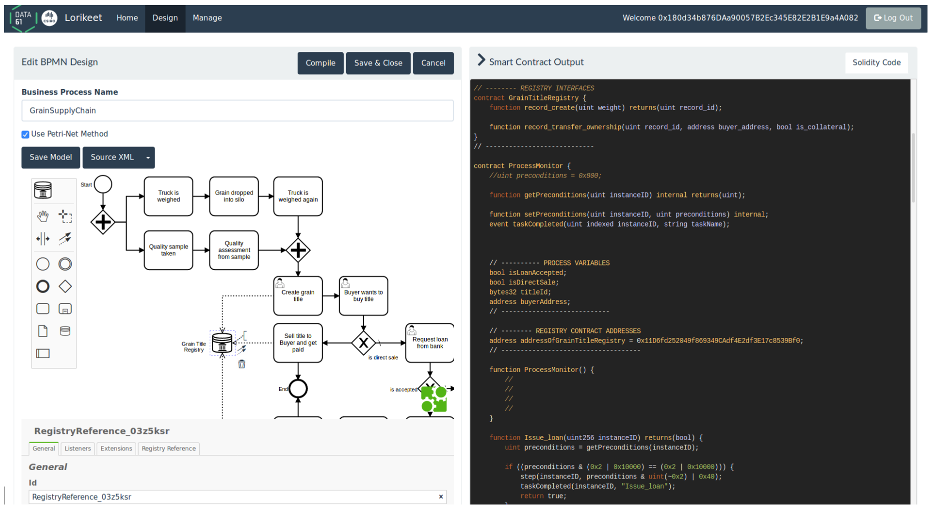
2 Tool Description

Fig. 1 illustrates the architecture of Lorikeet, which consists of a modeller user interface (UI), and three back-end components including BPMN translator, Registry generator and blockchain trigger. The modeller UI component is presented as a web application for users to build business process and registry models. The business process modeller is extended from the bpmn-js modelling library[3]. Business processes are modelled in the Business Process Model and Notation (BPMN) 2.0, while registries are modelled as in Regerator [7].

We extend BPMN 2.0 to support representation of and interaction with registries in the BPMN process model. The extension comprises two new elements, RegistryReference and ActionInvocation, which consist of new graphical notations and new XML attributes. A RegistryReference represents an asset data store on a blockchain while an ActionInvocation shows the asset registry action to invoke. On the registry side, the registry modeller provides a form for users to fill in the registry model input. The registry model consists of four parts, including basic information (registry name, description, user-defined data fields and their types), registry type (single or distributed), basic operations (Create/Read/Update/Delete and existence checking for each record) and advanced operations (record lifecycle management and foreign key). Specifically, we propose an access control policy to be enforced on registry record lifecycle management actions, such as `create, update, delete` and `transfer_ownership`. We provide an option to allow individual registry record lifecycle actions to be managed by a process instance. For an action invocation from the process instance to the registry, changes to the registry record are finalised only after the execution of process step logic is completed: the process instance smart contract calls the registry contract function. Since registry actions could fail, there is a checkbox on Lorikeet's user interface for atomic behaviour across registry actions and business process tasks: if marked as atomic, the registry update and the process state change either both fail or succeed.

Back-end components, including BPMN translator, registry generator, as well as blockchain trigger, are built adhering to a microservice-based architecture and deployed independently as Docker containers. The BPMN translator can automatically generate smart contracts in Solidity from BPMN models while the registry generator creates Solidity smart contract based on the registry models. The BPMN translator takes an existing BPMN business process model as input and outputs a smart contract. This output includes the information to call registry functions and to instantiate and execute the process model. The registry generator takes data structure information and registry type as fields, and basic and advanced operations as methods, from which it generates the registry smart contract. Users can then deploy the smart contracts on blockchain. The trigger communicates with an Ethereum blockchain node, and handles compilation, deployment and interaction with Solidity smart contracts.

[3] https://github.com/bpmn-io/bpmn-js



Business Process Name
GrainSupplyChain
☑ Use Petri-Net Method

RegistryReference_03z5ksr

General
Id
RegistryReference_03z5ksr

```
// -------- REGISTRY INTERFACES
contract GrainTitleRegistry {
    function record_create(uint weight) returns(uint record_id);

    function record_transfer_ownership(uint record_id, address buyer_address, bool is_collateral);
}
// --------------------------

contract ProcessMonitor {
    //uint preconditions = 0x800;

    function getPreconditions(uint instanceID) internal returns(uint);

    function setPreconditions(uint instanceID, uint preconditions) internal;
    event taskCompleted(uint indexed instanceID, string taskName);



    // ---------- PROCESS VARIABLES
    bool isLoanAccepted;
    bool isDirectSale;
    bytes32 titleId;
    address buyerAddress;
    // --------------------------

    // -------- REGISTRY CONTRACT ADDRESSES
    address addressOfGrainTitleRegistry = 0x11D6fd252049f869349CAdf4E2df3E17c8539Bf0;
    // ------------------------------------

    function ProcessMonitor() {
        //
        //
        //
        //
    }

    function Issue_loan(uint256 instanceID) returns(bool) {
        uint preconditions = getPreconditions(instanceID);

        if ((preconditions & (0x2 | 0x10000) == (0x2 | 0x10000))) {
            step(instanceID, preconditions & uint(~0x2) | 0x40);
            taskCompleted(instanceID, "Issue_loan");
            return true;
```

Fig. 2. Graphical interface of Lorikeet.

The BPMN and registry modeller UI interacts with the back-end microservices via an API gateway. The API gateway forwards API calls from the modeller UI, such as translating BPMN model to Solidity, to the corresponding microservice. Fig. 2 shows the business process modeller UI of our tool. Once the user makes changes to the BPMN model on the left-hand side, it is translated to the corresponding Solidity smart contract code which is displayed on the right-hand side. The registry modeller UI is similar to the business process modeller UI. On the left-hand sides is a form collecting the customised registry information.

3 Maturity, Demo, and Screencast

Lorikeet is a well-evaluated tool that is used for creating blockchain smart contracts in industry and academia. Besides the usual testing, we have used Lorikeet in collaborations with academics internationally and in student projects. We also have completed a number of external blockchain application projects using Lorikeet and received positive feedback from our clients.

For the demo, we will use Lorikeet to model a *grain title transfer process* use case [6] and generate smart contract code based on the models. We encountered this use case in our projects and discussions with industry, and modelled it based on publicly available information. Fig. 3 shows a simplified *grain title transfer process* modelled using Lorikeet. There is only one grain title registry interacting with this selected *grain title transfer* process.

A screencast demonstrating the usage of the Lorikeet tool can be found at https://drive.google.com/open?id=1rpy-oHbDVkXa6u4Fn73wSX8rINn1sv3U .

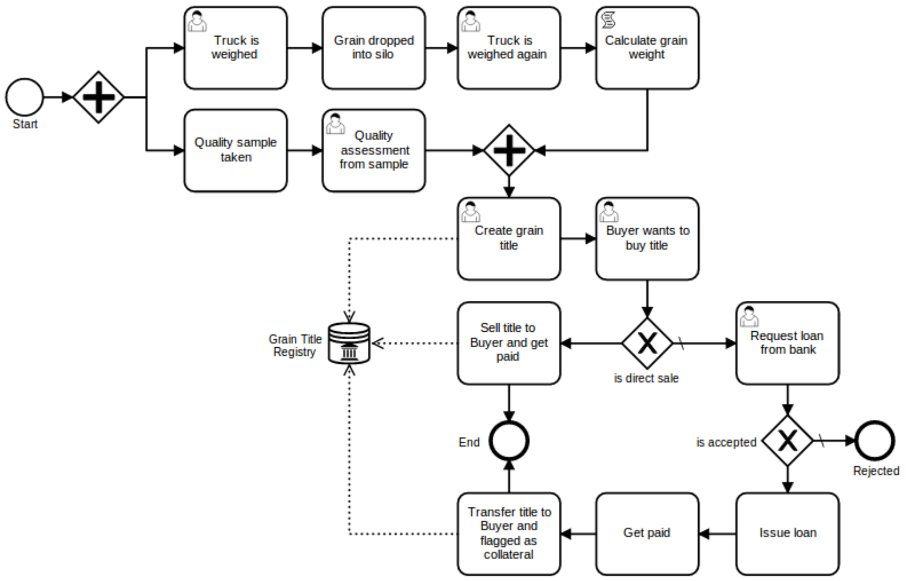


Fig. 3. Grain title transfer process model built using Lorikeet.

References

1. Distributed ledger technology: beyond blockchain. Technical report, 2016. UK Government Chief Scientific Adviser.
2. L. García-Bañuelos, A. Ponomarev, M. Dumas, and I. Weber. Optimized execution of business processes on blockchain. In *BPM'17: International Conference on Business Process Management*, Sept. 2017.
3. O. López-Pintado, L. García-Bañuelos, M. Dumas, and I. Weber. Caterpillar: A blockchain-based business process management system. In *BPM'17: International Conference on Business Process Management, Demo track*, Sept. 2017.
4. G. P. Release. Gartner survey reveals the scarcity of current blockchain deployments, May 2018. , accessed: 4 May 2018.
5. D. C. Schmidt. Guest editor's introduction: Model-driven engineering. *Computer*, 39(2):25–31, Feb 2006.
6. M. Staples, S. Chen, S. Falamaki, A. Ponomarev, P. Rimba, A. B. T. I. Weber, X. Xu, and J. Zhu. Risks and opportunities for systems using blockchain and smart contracts. Technical report, Sydney, 2017. Data61(CSIRO).
7. A. B. Tran, X. Xu, I. Weber, M. Staples, and P. Rimba. Regerator: a registry generator for blockchain. In *CAiSE'17: International Conference on Advanced Information Systems Engineering, Forum Track (demo)*, June 2017.
8. I. Weber, X. Xu, R. Riveret, G. Governatori, A. Ponomarev, and J. Mendling. Untrusted business process monitoring and execution using blockchain. In *BPM'16: International Conference on Business Process Management*, Sept. 2016.