

Design and development of Smart Contracts for E-government through Value and Business Process Modeling

Cristian Gómez, Francisco J. Pérez-Blanco, Juan Manuel Vara, Valeria De Castro, Esperanza Marcos
Kybele Research Group, Universidad Rey Juan Carlos, Madrid, Spain
{cristian.gomez, francisco.perez, juanmanuel.vara, valeria.decastro, esperanza.marcos}@urjc.es

Abstract

Administrations are constantly making great efforts to cope with the right of citizens to high-quality e-Government services. Since the advent of blockchain and smart contracts, they are trying to incorporate these technologies into their service offerings, which results in a complex task due to their nature and their inherent complexity. On the other hand, business and business process modeling facilitates the understanding and agreement between the different parties involved in the design of e-government services. All this given, this work introduces a model-based proposal to ease the integration of smart contracts into e-government services.

1. Introduction

Many of the economic efforts of governments in the last years are oriented towards the implementation or improvement of their digital platforms [29]. These platforms, which are collectively referred as “electronic government” or “e-government” are actually the result of the deeper introduction of Information and Communication Technologies (ICT) in society [12].

On the other hand, one of the most thrilling advances of ICT in the last years is probably the conception of the blockchain and the advent of smart contracts. These digital contracts are similar in nature to physically drafted contracts, with the advantage that they allow automatic verification of the conditions agreed in a transaction, without requiring the intervention of a third party acting as intermediary [8].

Due to the advantages offered by this technology, many governments have considered the re-designing of their services to integrate blockchain technologies in the processes that make up their services [19]. However, given the incipient nature of this

technology, these initiatives have encountered several problems such as: collection of legal aspects, security, ease of implementation in specific contexts, etc. [8][12][19], which together with those inherent to the (re-)designing of e-government services [13][14] result in a complex scenario.

Regarding the later, Business Process Modeling Notation (BPMN), as a kind of lingua franca for process modeling, has proven to be satisfactorily used for e-government service designing purposes [5]. We firmly believe not only process, but also organizational and value models can be used as well to facilitate the understanding between the different stakeholders involved in the designing of e-government services and thereby, serve as a basis to build consistent, realistic, and effective e-government services integrating blockchain technologies.

All this given, this work introduces a methodological and technical proposal to facilitate the design and development of smart-contract based services in the context of e-government. To that end, a model-based domain specific language (DSL) for the development of Solidity smart contracts has been developed and the relationships between smart contracts and value and BPMN models have been identified. We consider that the ability to semi-automatically explore these relationships by means of model-based technological bridges would facilitate the integration of smart contracts in the e-services offered by government administrations. Finally, model-based technological bridges will be built between business and business process modeling notations and smart contracts models exploiting these relationships through techniques based on Model Driven Engineering [18]. To that end, we will integrate the newly-developed DSL into modeling toolkit for Service Design [21] that supports different business and business process modeling notations, namely Canvas Business Model [20], e³value [10], Service Blueprint [25], Process Chain Network (PCN) [24] and BPMN [3].

The rest of this paper is organized as follows: Section 2 discusses briefly the use of smart contracts in the context of e-government. Section 3 introduces the methodological proposal to introduce smart contracts in the design of e-government services. Section 4 presents the technological solution supporting such proposal, while Section 5 uses a case study case to illustrate the relevance of the proposal. Finally, Section 6 summarizes relevant related works and Section 7 concludes by providing directions for further work.

2. Adopting blockchain and smart contracts in e-government

Numerous governments have used different digital tools to facilitate both the government administrations and citizens dealing with the processes that make up the services offered [12]. On the other hand, blockchain became a very popular term due to the boom of cryptocurrencies. In a nutshell, the blockchain is a distributed network formed by a series of nodes, which record each of the transactions that take place in such network in a chain of blocks known as “ledger” [4]. This digital ecosystem provides a high degree of security and trust to the different actors that use it, due mainly to the transparency of the actions taking place in the network.

More recently, one of the more appealing features of such network has been said to be its utility as a computation platform for smart contracts. A smart contract is a software program hosted on a blockchain that integrates a series of clauses (similar to a traditional one) that are automatically executed when the conditions included in the contract are fulfilled [6]. These digital contracts can be seen by governments as a transposition of the contracts or forms with which citizens interact to request some of the services offered through any administration, such as tax payment [29].

The most relevant characteristics of smart contracts for their integration in the services offered to their citizens are summarized as follows [12]:

- Transparency: all the transactions that take place are recorded within the accounting book or ledger, so it is easier to “track” these transactions than it is in other digital ecosystems with similar purposes.
- Automation: smart contracts allow businesses to automatically trigger commercial actions based on predefined conditions. This will boost efficiency by streamlining processes, helps to avoid possible frauds and reduce compliance and time costs.

- Objectivity: being a piece of software, it does not leave space to misinterpretation of the contract conditions, a common issue with traditional contracts.
- Immutability: once deployed on the blockchain, contracts cannot be modified.

Next, some examples of areas where these contracts have started to be used follows [4]:

- Intellectual Property: with the delivery of digital diplomas by Massachusetts Institute of Technology (MIT) or the University of Cyprus.
- Food: companies such as Carrefour or Walmart use these contracts to monitor the food distribution process from its origin to its sale.
- Health: proposals such as MedRec (Israel) for recording patient data.
- Administration: user voting management systems, such as FollowMyVote or birth registration in Illinois.

Despite the advantages of this technology, the adoption of the blockchain ecosystem does not escape from a series of drawbacks [1][2][12]:

- Immutability: being mentioned previously as one of the most outstanding advantages of smart contracts, it is also a major deficiency due to the inability to change the contractual terms of an agreement between various parties.
- Security: smart contracts do not escape their own nature. As software programs, they may contain security flaws derived from their design or implementation. Individuals could therefore take advantage from the immutability of the contract to exploit existing security flaws.
- Hardware dependency: it is a demanding technology in terms of resources needed to run it. This makes it difficult to establish and use it in developing countries or areas.
- Contamination: high energy consumption from miners constitutes a handicap for non-developed countries suffering problems with electricity supply and maintenance.
- Complexity: compared to others with higher levels of acceptance, it is still an incipient technology. Therefore, smart contract developers face a steep learning curve. The lack of tools to streamline development processes is indeed one of the most popular demand in existing literature [2].

3. Methodological Proposal

Given the inherent complexity of designing and implementing e-government services based on the use of smart contracts, this section introduces a model-based proposal to do so. The use of models

allows to abstract from the complexity of the underlying code, thus enabling non-technical people to think about contracts, and even negotiate their terms in the context of the targeted service.

3.1. Bridging smart contracts and service design

Figure 1 provides a conceptual overview of the proposal introduced in this paper. The underlying idea is that the definition of process (as BPMN) or exchange value (as e³value) models results way more intuitive for stakeholders as an intermediate step to the specification of smart contracts. These notations, along with Canvas, Service Blueprint or PCN, are commonly used when designing or redefining a service or e-service. There is even a modeling toolkit supporting these notations and bridging some of them [21]. Therefore, the models used to design the service, can be used as input to develop the smart contracts needed to implement the service.

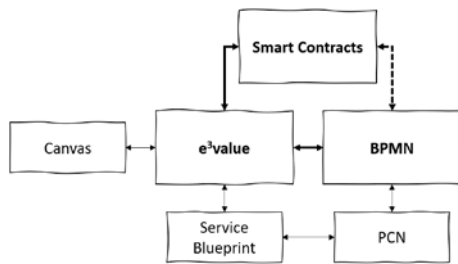


Figure 1. Generating smart contracts from high-level models

As next subsection explains, relationships among smart contracts and e³value have been analyzed as well. The e³value notation is used to reflect the value exchanges that occur between actors in a given context [10]. As a result, a number of mapping rules to generate a smart contract skeleton from an e³value model have been identified. The code generated will need manual refinement, but the proposal reduces drastically the effort needed to develop the contract. In addition, it enables business professionals to think, negotiate and discuss in terms of (smart) contracts.

This is obviously a two-way mechanism: partial versions of an e³value can be generated from a smart contract. This will help also when an implemented smart contract must be presented to business managers, since the model will provide a global and high-level overview of the contract and the underlying interactions. Finally, the relationships between BPMN and smart contracts are also being analyzed in order to support the same functionality in the mid-term.

The election of e³value and BPMN is not casual: the value exchanges collected in e³value models are

easily mapped into the interactions represented in a smart contract. On the other hand, BPMN serves to represent the processes in which the value exchanges take place, reflecting the interactions and messages exchanged between actors.

Note however that some other techniques for service design have also been integrated in the proposal. While this work focuses on the functionality related to smart contracts development, the same principles have been exploited in order to allow generating business and business process models from models expressed with other notations. As a result, the starting point to generate a smart contract could be not only an e³value model, but any of the remaining models supported by the proposal (Canvas business model, Process Chain Network or Service Blueprint). This way, a chain of model transformations [7] will serve to move forward from the starting model selected either to an e³value or BPMN one, and then to the smart contract.

On the other hand, e-service design is favored by the fact that business process models gathering the details of the service provisioned can be generated from the business models representing an overview of the organization and the context in which it performs its activity. This way, a Canvas business model could be the starting point to generate a Service Blueprint, a PCN or a BPMN model. This as well to facilitate the communication between the different stakeholders, since they could use their preferred modeling technique which would be easily and automatically converted into models expressed with other notations. As mentioned before, service design for e-government usually involves several stakeholders, each of them with its own interests and preferences.

3.2. Correspondences between e³value and smart contracts

Existing proposals analyzing the relationships between business process models and smart contracts such as Lorikeet [26] or Caterpillar [16]. However, we have found that there is a more direct correspondence with e³value due to the nature of the value exchanges it represents. Solidity is a programming language, similar to Javascript, for defining smart contracts in Ethereum net. Smart contracts in solidity have the following relevant aspects:

- A data type (address objects) to represent the different actors that will interact with the contract.
- A series of events that warn of the start or end of an action.

- A series of functions, which execute the defined code based on the logic programmed within them. It is in them, where the economic

transactions (exchange coupons, assets, coins, permission, etc.) in the contracts take place.

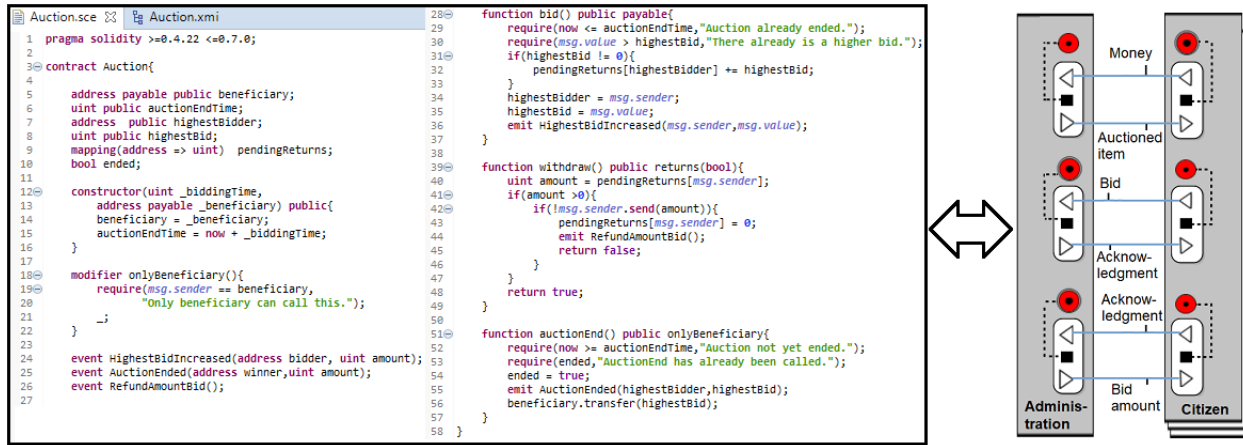


Figure 2. Correspondences between an auction smart contract and the correspondent e³value model

The example shown in Figure 2 is intended to illustrate the relationships found to hold between e³value and smart contracts. The left-hand side of the figure shows an auction smart contract coded in Solidity with the IDE developed as part of the proposal (see Section 4) while the right-hand side shows an e³value model, also elaborated with the developed tooling.

- e³value *Actors* (Citizen and Administration) are entities that carry out activities, granting them some type of benefit or corresponding benefit. These actors correspond to *Address* objects of in Solidity. These objects represent unique addresses, commonly associated with the different actors interacting with the smart contract.
- e³value *value ports* are related to the actions of receiving and sending the exchanged value objects. These ports are connected through *value exchanges* element to indicate the direction of the value exchange. Value ports and value exchanges could be then associated with the smart contract's functions. These functions execute the code for sending and receiving currency, services or products by the actors interacting with the contract.
- e³value *value objects* correspond to the assets sent and received by the actors in the contract. In terms of Solidity we will be talking about coupons, crypto assets, permissions, etc.
- e³value actors own *value interfaces* that group value ports to reflect the exchanges that can take place between actors. This grouping relationship between value interfaces and value ports can be mapped to the relationship that holds between

smart contracts themselves, which group together the functions encoded.

- e³value *stimulus events* (start and end) reflect the triggering or completion of a task. These events can be mapped into Solidity Event objects.

All this given, Figure 2 illustrates these correspondences by using the mentioned Auction smart contract. First, the smart contract can be modelled as one or several value interfaces. This is one the decisions to be made by the domain expert. Next, each address object corresponds to an actor, whereas the event objects are related with the stimulus objects collected in the e³value model. The assets being the subjects of interest for the contract (the ether received by the bidder, the different acknowledgements, etc.) are mapped to value objects in the models, which are interchanged through the value port and value exchange objects mapping the contract functions.

The following table tries to reflect in a simple way the correspondence previously analyzed between elements of a smart contract and elements of an e³value:

Smart Contract element	e ³ Value element	e ³ Value graphic symbol
Address/Address payable object	Actor/Market segment	
Smart contract class	Value Interface/Value interfaces group	
Functions	Value ports & Value exchange	


Assets (notifications, coupons, permissions, ...)	Value Objects	[Advices,goods...]
Events	Start / End Stimulus	

Table 1. Correspondences between smart contract and e³value models

4. Technological solution

This section introduces the modeling toolkit in which the proposal described in this work is being implemented and integrated. Thereby, the technical details of the toolkit are first discussed to later focus on the construction and integration of a DSL for smart contracts development. Such DSL will be later bridged with the rest of DSLs bundled in the toolkit by means of model-based techniques, namely model transformations and model weaving.

4.1. Development process

We plan to integrate the proposal of this work into INNoVaServ: a modeling toolkit which integrate several notations and functionalities for service design [21]. Figure 3 illustrates the conceptual architecture of INNoVaServ. It basically consists of an Eclipse IDE where several plugins as well as a layer of functionalities connecting them have been integrated. Each plugin implements a visual DSL. They were initially built atop of Eclipse Modeling Framework (EMF)/Eclipse Graphical Modeling Framework (GMF) [11] following the guidelines sketched in [28] for the development of model-based tools atop of Eclipse¹.

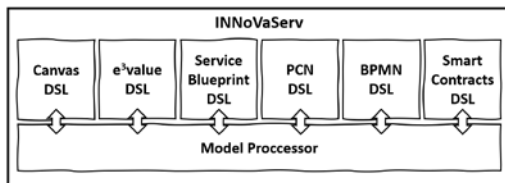


Figure 3. Simplified overview of INNoVaServ's conceptual architecture

However, due to the recent lack of GMF support, the migration of the DSLs from GMF to Sirius was addressed. Since Sirius is also based on Eclipse EMF, the development process of graphical editors is still similar to that of GMF.

On the other hand, the model processor layer supports several functionalities such as automatic model validation and fixing implemented with Acceleo. In addition, model transformations (M2M) among the different DSLs have been implemented

using the Epsilon family of languages [15]. Epsilon Transformation Language (ETL) supports many-to-many model transformations and it eases the combination of declarative rules with imperative constructions and lazy and greedy rules. This is an essential feature, since many of the model transformations developed are not direct but require certain level of interaction with the user to collect some design decisions that should guide the transformation, as it occurs when generating smart contracts from e³value models. In this sense, Epsilon Object Language (EOL) has been used to improve user interaction by means of dialog boxes and to handle the transformations accordingly. Also, each transformation generates a weaving model to materialize the relationships that hold between source and target models. The presented tool aims to reduce the digital gap between the different government professionals through the use of model transformations, facilitating the analysis of requirements and the implementation of the e-service in question, being one of the most requested demands [12][19].

To that end, Modelink, a simple but useful multi-panel editor provided by Epsilon is used. It consists of 2-3 side-by-side EMF tree-based editors, which allows visualizing the source and target models, along with the corresponding weaving model. Note that relationships collected in the latter can be directly edited in the editor.

Again, it is worth noting that the visualizations provided by Modelink are planned to be improved by developing ad-hoc multi-panel editors like those presented in [27]. For instance, integrated overviews of all the models involved in a given project and their relationships could be provided this way.

Finally, since the toolkit is still basically an EMF/GMF tool, it is consequently interoperable with any other EMF/GMF existing tool. Note that there exists plenty of them since EMF/GMF has turned to be the de-facto standard for the development of model-based tools for the last 10 years. For instance, leaning on Papyrus, UML models could be almost immediately combined with those supported by INNoVaServ for business management tasks.

4.2. SmaC: model-based tool support for smart contracts

First of all, SmaC² is a textual DSL that supports the coding of smart contracts with Solidity. These contracts can be injected to EMF models and then subject to any model-based processing task.

¹ Ecore (meta)models: <https://postimg.cc/gallery/0NLJ9PG>

² <https://cutt.ly/1g3BRdh>

SmaC has been developed with Xtext, a tool for the development of textual DSLs. From a grammar specification, Xtext generates an infrastructure for the textual DSL. This infrastructure is integrated by a series of projects acting as containers, either to house the metamodel from which SmaC EMF models can be created or to collect a series of functionalities encoded in Xtend that facilitate the elaboration of these textual models. One of the functionalities defined in SmaC is indeed to produce EMF models from Solidity SmaC contracts.

In relation to some of the challenges of coding smart contracts [1], SmaC presents a series of advantages detailed below:

- SmaC establishes a structural pattern for the coding of a smart contract. The specified smart contract is therefore made more readable and understandable by the developer.
- It requires defining a gas control when executing the loop actions. This avoids infinite loops which may lead to security issues.
- It contains a series of ad-hoc facilities that can be easily extended or modified at the user's request. Some of these facilities are code autocompletion, validation and quickfixes.
- A set of ad-hoc data types (User & Company) has been bundled in the DSL grammar to facilitate the correspondence between SmaC and e³value.
- Any SmaC model is itself a smart contract defined in Solidity. Therefore, it can be compiled to bytecode by any IDE intended for it, such as Remix.

In summary, SmaC provides a model-based IDE for the development of Solidity smart contracts which entails a number of current and potential advantages for developers.

5. Case study

This section presents a case study focused on the electronic auctions used by the Spanish Government to illustrate the proposal of this work to bridge service design notations and smart contracts for e-government. Please note that all the excerpts in this section correspond to models elaborated with the tooling support developed.

It should be noted that the example used in this case study corresponds to a real scenario which has been emulated in the laboratory. To that end, the guidelines sketched in [23] for conducting cases studies were followed to run a brief experiment with a group of 51 undergraduate students on Service Engineering with no much experience developing Smart Contracts but good knowledge in e³value. Note

that the main goal of this experiment was just to obtain some evidence to assess the viability of the proposal.

The first step in the case study design is to define the objectives and route plan. The main objective of this experiment was to evaluate how much of a Smart Contract model could be generated from an e³value one and vice versa. To that end, the models presented below were provided to the students, who were later asked to refine them in order to incorporate the use of Smart Contracts.

Regarding preparation for data collection phase, no survey or form were needed to reach our goal. The only data collected consisted of the models generated by the students. Once the resulting models were collected, the e³value's were compared with the correspondent Smart Contracts so that the number of elements with direct correspondence among these notations were counted. The percentage of Smart Contract elements generated from an e³value model and vice versa were then computed. When moving from Smart Contract to e³value models the transformation coverage was 48,15% on average. By contrast, the opposite direction reached a transformation coverage of 33,35% on average. Both are preliminary acceptable percentages given into account the differences between the correspondent models. The case study used in the experiment is described as follows.

As Figure 4 shows, the auctions consist currently of the publication of a series of goods (homes, vehicles, etc.) offered by the government administration through an electronic portal. These goods have been previously confiscated to debtors who failed to pay. Citizens authenticate themselves electronically in the portal and register their bids during a period previously established by government. In 3 to 14 days the administration sends these data to a judicial administration so that a public official can grant its proof of faith to both the administration and the bid participants on the winner. Once the result is issued, the winner has 40 days to make the payment effective, at the expense of the judicial administration granting the property title within a period of time not determined. To improve the understanding of the auction process, we have only modeled the process in which a participant bids and waits for the result of the auction. In case she wins the auction, she would have to pay the remaining fee. Otherwise, she must wait an indefinite period (days, weeks, months) for the government administration to reimburse her the amount offered in the auction.

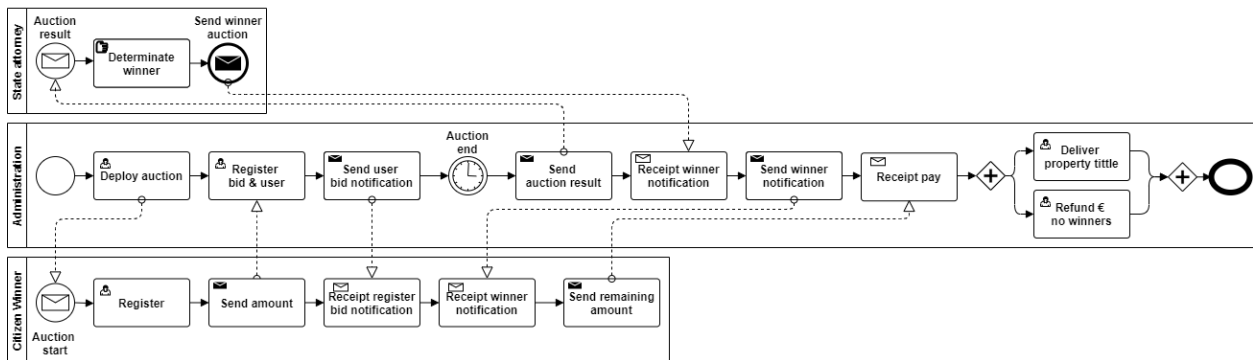


Figure 4. BPMN model excerpt: auction process WITHOUT smart contracts

Figure 5 illustrates the value exchanges taking place in the auction process currently run by the Spanish Government administration. In this case, the value objects exchanged between the citizens participating in the auction and the administration of the treasury when bidding are the amount of the bid and the notification of participation.

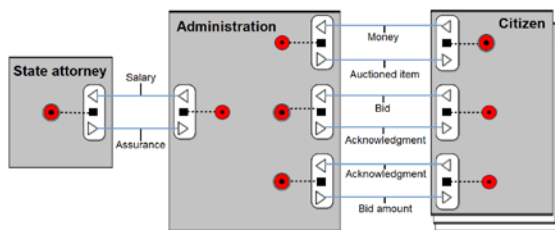


Figure 5. e³ value model excerpt: auction process without smart contracts

To deduct the winner, the administration requests a judicial administrator to validate the process, receiving from the administration his salary for the work carried out. Regarding the winning bidder, there is an exchange between the auctioned good and the bid amount. Regarding non-winning bidders, the administration refunds the money and the citizen issues a notification upon reception.

To improve the process and take advantage from the features of smart contracts mentioned in Section 2, a new process is proposed hereafter.

To that end, we lean on the smart contract presented in Figure 2, which corresponds to an electronic auction model in which the participants (each one represented unambiguously through an *address* field) interacts with a contract deployed on Ethereum to bid for the goods offered by the administration. It is indeed the administration who deploys the contract and establishes a period of time for the participants to bid through it. Participants bid must overcome the highest bid to date in order for the contract to record the bid and its unique address (*bid* function). Recall that such address acts as an

identifier for the participant does not win the auction and requests a refund. Once the auction period has expired, the government administration will invoke the *auctionEnd* function, so his account is reimbursed with the funds of the highest bid and issuing a notification that the auction has finished indicating the winning bid. The rest of participants can then request to have their currency instantly refunded to their account (*withdraw* function).

Figure 6 shows the changes needed in the BPMN model to run the auction process by incorporating smart contracts which entails basically the elimination of a third party and its corresponding tasks, which are now carried out by the smart contract. These changes are basically based on the Smart Contracts advantages analyzed in Section 2.

The use of smart contracts in this scenario brings a series of improvements regarding the original process illustrated by Figure 4:

- Elimination of intermediary actors thanks to the automation of tasks. The smart contract allows to eliminate the judicial administrator who intervened to certify the auction result, being carried out by the smart contract itself upon completion of the auction process.
- Reduction of time and monetary costs. By using a smart contract, the time lapse for sending the result to the judicial administration is 0, because the auction result is directly stored in the contract. As soon as the administration invokes the *auctionEnd* function, it obtains the funds stored in the contract which correspond to the highest bid instantly (as opposite to the former 40 days period). Likewise, as soon as they invoke the *withdraw* function, the funds are instantly returned to the non-winning participants.
- Automatic registration of all the bids made. Possible loss or alteration of information is

prevented by the smart contract while “transparency” is ensured.

- Complete control of the auction process. For instance, if a bidder is not able to fulfill the payment promise of his/her bid the auction process is suspended. The use of smart contracts

implies that bidders will prepay by depositing the funds along with their bid in the contract. Likewise, errors from the judicial administration which could also result in the auction being suspended are now avoided.

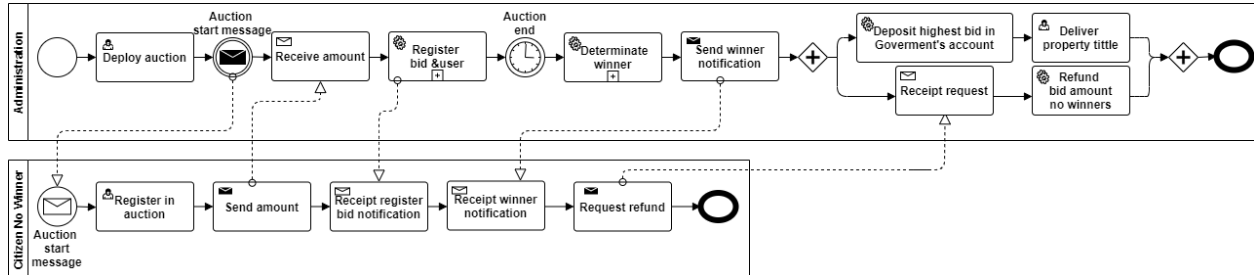


Figure 6. BPMN model excerpt: auction process WITH smart contracts

Additionally, as discussed in Section 3.1, all the capabilities of the modeling toolkit for Service Design can help to shorten the distance among the stakeholders involved in the provision of the service. For instance, the e³value model could be used to generate semi-automatically a partial view of the correspondent Canvas Business Model for the auction service. Likewise, regarding business process models, it is also possible to exploit the information gathered in the BPMN model to generate complete PCN models. Finally, if needed, it is also possible to reduce the level of detail in the process models used to design the service. This can be useful for displaying high-level service designs without delving into the details of service delivery but offering a simplistic and concise vision of the process. In this sense, it is possible to generate Service Blueprints from BPMN, PCN or e³value models. Note that Service Blueprint is indeed one of the most popular notations for Service Designers.

All in all, supporting this set of business and business process modeling notations will serve then to close the gap between IT experts, non-IT experts, and Operations Managements experts. One cannot stay that one notation is better or worse than the others. All of them are somehow useful in the sense that they can provide complementary points of view more oriented to each of the stakeholders involved in the context of e-government service design.

6. Related works

Due to the functionalities offered by smart contracts, they are increasingly being integrated as a solution to existing problems in several of the services offering of companies and government administrations. In order to ease and improve the integration of smart contracts in these processes,

proposals for the development and deployment of smart contracts through BPMN such as Lorikeet [26] or Caterpillar [16] have emerged. However, these proposals hinder the interpretation of the different actors that are part of the process when defining the smart contract, due to the digital gap that may exist between business professionals and developers. We have discovered however that e³value fits much better with the nature of smart contracts. There are in fact some works on the utility of e³value models as a way to identify potential business cases for the application of blockchain technologies [9] but they are far from identifying the similarities between the concepts on which value models and smart contracts are based. There are other proposals that aim to reduce the digital gap by adopting smart contracts through state machine such as FSolidM. However, this proposal is more focused on developers than on stakeholders involved in the provision of the service, unlike SmaC that includes both sectors [17].

On the other hand, there are DSL-based proposals to provide high-level languages for the definition of smart contracts [6][22]. Regarding them, SmaC, supports the generation of fully functional contracts and allows connecting them to the processes where the contracts should be integrated.

Regarding tool support for business and business process modeling, we have found no tool supporting all the notations currently supported by INNoVaServ. Likewise, there is no proposal currently exploiting the relationships between those notations as it happens with INNoVaServ.

There exist some tools supporting some of these notations, like Canvanaizer, Miro, draw.io, Lucidchart or Gliffy, which are web-based applications that supports (some of them) collaborative edition of Canvas Business Model, Service Blueprint or BPMN. They have simple and

intuitive graphical interfaces, but they are not model based tools, so subsequent processing of the information collected in those models is not contemplated. Likewise, they lack export support with suitable formats for post-processing (like XML), the output format being a simple image in most cases. Some of them are commercial solutions offering free limited editions, while others are completely free.

All this given, to the best of our knowledge this is the first proposal to consider the business and business process modelling notations discussed here along with smart contracts, and providing tool support to use them in the context of e-government service design.

7. Conclusion and future work

Smart contracts and its ecosystem offer several advantages to improve the services offered by e-government administrations, like removal of unnecessary activities or intermediaries, transparency, cost reduction or confidence increase.

The integration of smart contracts into these services implies however a re-designing of services and some mechanisms to shorten the distance between smart contracts (software pieces) and business experts.

Given the incipient nature of smart contracts and the complexity associated to their development, if they were to be integrated in service design activities, a high-level easy way of dealing with and think about them was needed.

In this sense, the main contribution of this work is to integrate a series of business and business process modeling notations with a model-based solution for smart contracts development. The resulting toolkit bundles a set of DSLs and will support the partial generation of smart contracts from e³value of BPMN models and vice versa.

To that end, the relationships that hold among the main elements of e³value and smart contracts have been analyzed. These relationships are being implemented by means of unidirectional model transformations to support the generation of Solidity code skeletons from e³value models and vice versa, the generation of e³value models from smart contracts. As this functionality is integrated in the modeling toolkit, it will be possible to ease the integration of smart contracts in service design activities from the early stages of the process.

Regarding the scope of the proposal, there are however a series of limitations that we plan to address as follows:

- As mentioned in Section 5, so far the proposal has been validated by means of a case study run

in the laboratory. From this experiment we obtained some preliminary metrics to assess the quality of the work, as the percentage of elements generated by the model transformations currently bundled in the tool. Nevertheless, a formal validation, probably in the shape of a real project, is needed.

- Another limitation is related with the number of correspondences identified between Smart Contract and e³value models. Interactive techniques to exploit potential correspondences will serve to improve the proposal.
- Note also that it is not possible to generate complete smart contracts from an e³value model. Instead, it is possible to generate a skeleton, i.e. class headers, methods, etc. but not their content. Some kind of user interactive assistant allowing manual completion of code when moving from an e³value model to a smart contract would definitively help to improve the proposal.
- Finally, the technological solution presented here is built atop of Eclipse, what entails a number of complications related with plug-in dependencies, installation issues, etc. The current trend in this sense is to deploy the tools in the cloud (see the latest works from Obeo for instance³). We plan to address this movement in the mid-term.

Regarding next steps of this work, given that both the modeling toolkit and the DSL for smart contracts are already running, we are already working to exploit the relationships found between e³value and smart contracts to provide automatic transformations between them and bundle them into the toolkit. In the medium term, relationships with BPMN and other business modeling techniques will be exploited as well. Likewise, a block-based visual concrete syntax for the smart contracts DSL is being developed.

Acknowledgements. This work has been partially funded by the Regional Government of Madrid, through the FORTE-CM project (S2018/TCS-4314) and the Spanish MINECO, through the MADRID project (TIN2017-88557-R).

³ <https://www.obeo.fr/en/products/554-product-obeo-cloud-platform>

References

- [1] Alharby, M., & van Moorsel, A. (2017). A systematic mapping study on current research topics in smart contracts. *International Journal of Computer Science & Information Technology*, 9(5), 151-164.
- [2] Bosu, A., Iqbal, A., Shahriyar, R., & Chakraborty, P. (2019). Understanding the motivations, challenges and needs of blockchain software developers: A survey. *Empirical Software Engineering*, 24(4), 2636-2673.
- [3] Chinosi, M., & Trombetta, A. (2012). BPMN: An introduction to the standard. *Computer Standards & Interfaces*, 34(1), 124-134.
- [4] Crosby, M., Pattanayak, P., Verma, S., & Kalyanaraman, V. (2016). Blockchain technology: Beyond bitcoin. *Applied Innovation*, 2(6-10), 71.
- [5] Delgado, A., Calejari, D., González, L., Montarnal, A., & Bénaben, F. (2020, January). Towards a Metamodel Supporting E-government Collaborative Business Processes Management within a Service-based Interoperability Platform. In *Proceedings of the 53rd Hawaii International Conference on System Sciences*.
- [6] Frantz, C. K., & Nowostawski, M. (2016, September). From institutions to code: Towards automated generation of smart contracts. In *2016 IEEE 1st International Workshops on Foundations and Applications of Self* Systems (FAS* W)* (pp. 210-215). IEEE.
- [7] Garcés, K., Vara, J.M., Jouault, F. et al. Adapting transformations to metamodel changes via external transformation composition. *Softw Syst Model* 13, 789–806 (2014).
- [8] Giancaspro, M. (2017). Is a 'smart contract' really a smart idea? Insights from a legal perspective. *Computer law & security review*, 33(6), 825-835.
- [9] Gordijn, J., & Akkermans, H. (2001). Designing and evaluating e-business models. *IEEE intelligent Systems*, (4), 11-17.
- [10] Gordijn, J. E-business value modelling using the e3-value ontology. In W.L. Curry editor, *Value creation from e-business models*, pp. 98--127, Oxford, UK, 2004.
- [11] Gronback, R. C. (2009). Eclipse modeling project: a domain-specific language (DSL) toolkit. Pearson Education.
- [12] Hou, H. (2017, July). The application of blockchain technology in E-government in China. In *2017 26th International Conference on Computer Communication and Networks (ICCCN)* (pp. 1-4). IEEE.
- [13] Juell-Skielse, G., & Perjons, E. (2009, July). Improving E-government through benefit analysis and value modeling. In *2009 33rd Annual IEEE International Computer Software and Applications Conference (Vol. 1)*, pp. 332-339. IEEE.
- [14] Jussila, J., Sillanpää, V., Lehtonen, T., Helander, N., & Frank, L. (2019, January). An activity theory perspective on creating a new digital government service in Finland. In *Proceedings of the 52nd Hawaii International Conference on System Sciences*.
- [15] Kolovos, D. S., Paige, R. F., & Polack, F. A. (2008, July). The epsilon transformation language. In *International Conference on Theory and Practice of Model Transformations* (pp. 46-60).
- [16] López-Pintado, O., García-Bañuelos, L., Dumas, M., & Weber, I. (2017, September). Caterpillar: A Blockchain-Based Business Process Management System. In *BPM (Demos)*.
- [17] Mavidrou, A., & Laszka, A. (2018, February). Design secure ethereum smart contracts: A finite state machine based approach. In *International Conference on Financial Cryptography and Data Security* (pp. 523-549). Springer, Berlin, Heidelberg.
- [18] Mellor, S.J., Clark, T., & Futagami, T. (2003). Model-driven development: guest editors' introduction. *IEEE software*, 20(5), 14-18
- [19] Ølnes, S. 2016. Beyond Bitcoin enabling smart government using blockchain technology. *Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics)*. Springer, Cham. 253–26.
- [20] Osterwalder, A., Pigneur, Y., Oliveira, M. A. Y., & Ferreira, J. J. P. (2011). *Business Model Generation: A handbook for visionaries, game changers and challengers*. *African journal of business management*, 5(7), 22-30.
- [21] Pérez-Blanco, F. J., Vara, J. M., Gómez, C., De Castro, V., & Marcos, E. Model-Based Tool Support for Service Design. In *International Conference on Fundamental Approaches to Software Engineering* (pp. 266-272), 2020.
- [22] Regnath, E., & Steinhorst, S. (2018, September). SmaCoNat: Smart Contracts in Natural Language. In *2018 Forum on Specification & Design Languages (FDL)* (pp. 5-16). IEEE.
- [23] Runeson, P., Höst, M. Guidelines for conducting and reporting case study research in software engineering. *Empir Software Eng* 14, 131 (2009).
- [24] Sampson, S.E. (2012). Visualizing service operations. *Journal of Service Research*, 15(2), 182-198.
- [25] Shostack, G.L.: Designing services that deliver. *Harvard Business Review* 62(1) (January 1984) 133–139.
- [26] Tran, A. B., Lu, Q., & Weber, I. (2018). Lorikeet: A Model-Driven Engineering Tool for Blockchain-Based Business Process Execution and Asset Management. In *BPM (Demos)*. (pp. 56-60).
- [27] Vara, J. M., Bollati, V. A., Jiménez, Á., & Marcos, E. (2014). Dealing with traceability in the MDD of model transformations. *IEEE Trans. on Software Engineering*, 40(6), 555-583.
- [28] Vara, J. M., & Marcos, E. (2012). A framework for model-driven development of information systems: Technical decisions and lessons learned. *Journal of Systems and Software*, 85(10), 2368-2384.
- [29] Yang, L., Elisa, N., & Eliot, N. (2019). Privacy and Security Aspects of E-Government in Smart Cities. *Smart Cities Cybersecurity and Privacy*, 89–102.