Towards a Service-Oriented Architecture for eVoting

Boris Shishkov[1,2] and Marijn Janssen[3]
[1]*Institute of Mathematics and Informatics, Bulgarian Academy of Sciences, Sofia, Bulgaria*
[2]*IICREST, 53 Iv. Susanin Str., Sofia, Bulgaria*
[3]*TBM – TU Delft, Jaffalaan 5, Delft, The Netherlands*
b.b.shishkov@iicrest.org, M.F.W.H.A.Janssen@tudelft.nl

Keywords: eVoting, Conceptual Model, Requirements, Architecture.

Abstract: The latest advances in the Information and Communication Technology (ICT) are changing our society, but have different implications on different domains. Some domains, such as the digital content –based businesses, are enjoying (almost) full ICT utilization whereas other domains, assuming physical and/or societal and/or "intuitive" inputs, are much less successful in terms of digitization. Voting using digital technology (or "eVoting" for short) is in between those domains since: (i) the mere process of voting is a very good "candidate" for digitization but at the same time (ii) the "surrounding" societal aspects are often difficult to "frame" as Internet-based services. (i) can be seen from the "voting through computer" observed in several European countries while (ii) can be seen from the lack (to date) of technology-enabled systems completely supporting the voting process and its related aspects. Further, the conceptualization and implementation of any voting system is to originate from legislation – this makes the goal of resolving (i) + (ii) even more challenging. Hence, to benefit from ICT, the question remains what should be done and how it should be done. The step from legislation to requirements and implementations taking into account socio-technical aspects, is crucial for the successful realization of eVoting. Despite its relevance, this has been given hardly sufficient attention in literature. This void is addressed by the current position paper; the contribution of the paper is two-fold: we firstly propose a general technology-independent conceptual model on voting and on this basis, we propose requirements for (partially) digitizing this process. Requirements are dependent on the societal context and therefore we opted for focusing on one particular EU country where the transition to eVoting is currently under discussion. We have planned as future research to reflect the identified requirements into architectures and implementations, and to get experts' feedback on this.

1 INTRODUCTION

Advances in the Information and Communication Technology (ICT) are changing our society, and each domain has its own "way" of utilizing computer / Internet –based services. Some domains, such as the digital content –based businesses, are enjoying (almost) full ICT utilization. Other domains, assuming physical and / or societal and / or "intuitive" inputs, are less successful in terms of digitization. Technology-enabled voting (referred to as "eVoting" in the current paper) is in between those domains since: (i) the mere process of voting is a very good "candidate" for digitization but at the same time (ii) the "surrounding" societal aspects are often difficult to "frame" as Internet-based services. (i) can be seen from the "voting through computer" observed in several European countries while (ii) can be seen from the lack (to date) of a technology-enabled system completely supporting the voting process and its related aspects (Scammell, 2016). Hence, the eVoting domain should be conceptualized from a socio-technical perspective. Further, the conceptualization and implementation of any voting system is to originate from legislation – this makes the goal of resolving (i) + (ii) even more challenging. Thus, the question remains what should be done and how it should be done, in order to both benefit from ICT advances and stay adequate in terms of societal integrity. Meeting all requirements is crucial for e-Voting, as failure in voting is not an option in a democratic society. Still, the step from legislation to requirements and implementations is essential for the successful realization of eVoting. Nevertheless, despite its relevance, this particular step has been given hardly sufficient attention in literature. This justifies our work, reported in the current position paper – to conceptualize the voting process and derive

Shishkov B. and Janssen M.
Towards a Service-Oriented Architecture for eVoting.
DOI: 10.5220/0006223601870195
In *Proceedings of the Sixth International Symposium on Business Modeling and Software Design (BMSD 2016)*, pages 187-195
ISBN: 978-989-758-190-8

requirements (and guidelines) for eVoting. Hence, the contribution of the paper is two-fold: we firstly propose a general technology-independent conceptual model on voting (by "voting" we mean the political voting for parliament, president, mayor, and so on, taking place in different countries, as opposed to for example corporate voting for board of directors or other kinds of voting) and on this basis, we propose requirements for (partially) digitizing this process. Still, since those issues are inevitably based on a particular societal context and because we need to be "concrete" in our modeling activities, we have decided to base our work on the situation in one particular EU country where the transition to eVoting is currently under discussion. We have planned as future research to reflect the identified requirements into architectures and implementations, and to get experts' feedback on this.

In the remaining of this paper: In Section 2 we present theoretical background and in Section 3 we present our research focus and propose a conceptual model on voting accordingly. On this basis, we derive requirements (Section 4). Then, in Section 5, we discuss the next step, namely reflecting the identified requirements in architectures and implementations, and we propose some general guidelines in this regard. Finally, we present the conclusions in Section 6.

2 THEORETICAL BACKGROUND

We propose a technology-independent model on voting, based on the theories of *LAP – Enterprise Ontology* (Dietz, 2006), *Organizational Semiotics* (Liu, 2000), *Workflow Management* (Van der Aalst, 2011), *Service-Oriented Computing* (Papazoglou, 2012), and *Conceptual Modeling* (Insfran et al., 2002). Those are briefly outlined in the current section.

2.1 LAP – Enterprise Ontology

The *Language-Action Perspective* (*"LAP"*) theory (Shishkov et al., 2006) emphasizes the importance of interaction and communication, recognizing that language is not only used for exchanging information, as in reports (for example), but that language is used also to perform actions, as in promises or orders (for example). Such actions are claimed to represent the foundation of communities and organizations / enterprises. This relates to the *white-box model* of an organization that is of key importance for building valid enterprise ontologies – this model acknowledges actors (the entities fulfilling corresponding actor-roles) who may be involved not only in *production acts* (for example: deliver a pizza) but also in *coordination acts* (for example: promise a delivery), and those acts may be of relevance to three perspectives of an organization, namely: *documental* (documents being created and used, for example), *informational* (customer enters PIN in order to realize a bank transaction, for example), and *essential* (the bank transaction itself, for example). Finally, *Enterprise Ontology* considers a generic interaction atomic pattern, claiming that any complex interaction can be decomposed in such pattern primitives and there are always two roles, namely customer (the one who initiates anything, for example – order something) and producer. There is a *request-promise-execute-state-accept* actions sequence between them and it can be reflected in a success layer and also a failure layer, as well as discussion layer, in between. For more information on Enterprise Ontology, interested readers are referred to (Dietz, 2006).

2.2 Organizational Semiotics

Organizational Semiotics (OS) addresses a number of concepts, such as *sign* and *affordance*, as essentially useful in modeling a (real-life) system and adequately considering relationships and meanings. Often what we observe goes beyond the primary "appearance" – for example, one could hold a Rolex pen not only as a means of writing but also as a way of demonstrating wealth (this is a sign). As for the affordance concept, it relates to potential abilities (for example: a book affords to be borrowed). Those concepts and also other OS concepts, allow for building complex models that reflect both *semantics* and *norms* (rules), and that is reflected in the widely popular OS norm pattern:

`whenever` <condition>
`if` <state>
`then` <agent>
`is` <deontic operator>
`to` <action>

The OS norm pattern is considered useful in modeling relationships among entities, in the context of a business process (Shishkov et al., 2006). For more information on OS, interested readers are referred to (Liu, 2000).

2.3 Workflow Management

It is claimed that any business process can be viewed

as a collection of processes, where a *process* can be described as "*a set of identifiable, repeatable actions which are some way ordered and contribute to the fulfilment of an objective*"; typical process patterns are *sequence*, *parallelism*, *split*, and so on.

Workflows play useful role in modeling business processes and those models can be enriched in terms of OS norms and / or entities information. For more information on Workflow Management, interested readers are referred to (Van der Aalst, 2011).

2.4 Service-Oriented Computing

Web services appear at high-level to be (dynamically) composed by users, hiding thus their underlying technical complexity – this complexity is with the software components who are implementing the corresponding service(s). *Composability*, *traceability*, and *interoperability* are hence of crucial importance in web service provisioning. More information on those issues can be found in (Papazoglou, 2012).

2.5 Conceptual Models

We consider a *conceptual model* as an abstraction with regard to the real world. As for information (IT) systems being developed, they inevitably need to be based on such abstractions because among other things, an information system is about the automation of real life processes. Nevertheless, the value of conceptual modeling efforts often remains unclear (Insfran et al., 2002) and often software engineers do not know whether a conceptual model represents the user's requirements. Finally, we argue that in order to be useful in such a context, a conceptual model is to adequately capture the *real life functionalities* that would be (partially) automated, as part of the development of information systems. In order to address this and taking especially an *eVoting perspective*, we:

a. would not keep the conceptual model too abstract (we would have it reflect a particular (voting) context);
b. would separate the technology-independent conceptual modeling from the IT-inspired requirements identification and specification;
c. would make particular assumptions in order to "take out" of consideration aspects with no relevance to what could actually depend on the IT system as such.

We argue that this would lead to establishing a more explicit role of conceptual modeling with regard to the development of information systems (particularly in the eVoting context) and also to guaranteeing that conceptual modeling is aligned to requirements engineering.

For this reason, we will firstly outline (in Section 3) the "modeling context", namely the situation (regarding eVoting) in one EU country (Sub-Section 3.1), secondly, we make our assumptions – in line with what was mentioned above (Sub-Section 3.2), thirdly, we propose a technology-independent model (Sub-Section 3.3), and we extend this, by identifying and specifying requirements (Section 4).

3 THE VOTING MODEL

As already mentioned, in this section, we consider the societal context, we present assumptions, and in the end – the technology-independent model on voting.

3.1 Societal Context

For the sake of "grounding" our research to a particular societal context / case, we went for considering a concrete eVoting context, namely the situation in *Bulgaria* (Konstantinov et al., 2009). The latest developments in Bulgaria with regard to eVoting are considered "representative" because they reflect the current EU visions on that issue, trying to balance between the scepticism (observed in Germany and other countries) and the "success stories" (observed in Estonia and other countries). Moreover, the current legislation changes and initiatives in Bulgaria reflect the latest ICT developments which was not the case in the countries who have introduced eVoting several years ago. For this reason, it is not surprising that eVoting dominated an October'15 referendum in Bulgaria (Plevneliev, 2016); then the majority of Bulgarians voted in support of "IT-enabled voting" and the "pro" campaign was backed by particular stated eVoting *public demands* that may (eventually) be reflected in corresponding legislation changes; some of those demands are presented and briefly discussed below:

- *secrecy of vote*, possibly achieved through anonymous credentials, such that not even the system "knows" how a person has voted;
- *cost adequacy*, possibly achieved through smart decisions rather than posh hardware that would generate future "dependencies";
- guarantee against *violations* with regard to the way the system works;
- guarantee against *manipulations* of the final

voting results;

- support of *secure communication* between the computers and the servers that is to be possibly cryptography-enriched.
- *controllability* - any third parties should be able to "verify" that the system is working properly;
- guarantee that each vote has been counted and that the person who had voted would *not be allowed to vote again*;
- *fault-reaction* is to be established as guarantee that even if the system (partially) crashes, it would recover and this would not affect its storage and processing functions;
- *ease of use* even by persons who are not of high computer literacy;
- no need for extra *qualification* of the election authorities.

Obviously, such a public demands list cannot be exhaustive and it is also inevitably unstructured. Moreover, those "demands" reflect a mixture of things – from FUNCTIONAL to NON-FUNCTIONAL, from CONCEPTUAL to TECHNICAL, and so on. For this reason, we should "extract" some information in support of our technology-independent functional conceptual model, and we should as well extract information in support of the requirements' identification to be addressed in Section 4. Hence, in the following list, we refine the demands, to achieve *input* for our conceptual model, abstracting from all non-functional and technical aspects:

- secrecy of vote;
- fair reflection of votes;
- fair communication of inter-mediary results;
- fair counting of votes;
- the voting – easy for all having rights to vote.

Thus, in line with those demands, any person with voting rights in Bulgaria should be able to vote. This assumes that no special qualification / skills are needed, in such a way that both his/her vote is fairly reflected in the voting system and the secrecy of his / her vote is guaranteed (this means that the vote should be counted but it should not be linked in any way to the person who has executed the voting). Finally, it should be guaranteed that all intermediary results are properly communicated to the voting "central" and in the end – all votes are correctly counted, to guarantee adequate political representativeness.

In Bulgaria, people vote for parliament, for president, for local authorities, and also in referenda. Voting is preceded by a campaign. During the campaign, candidates / parties have the right to register for the vote and also to campaign for / promote their ideas. This is supposed to stimulate people to analyse the current situation and the (potential) impact of different political influences. Then, the campaign stops and the day before the elections is made free of campaigning – to avoid pressure on voting in a certain direction. This is followed by the voting "day" lasting 12 hours; during this 12-hour period, people have the right to go and vote (or not to do so), and once voting is done, it is done – it is impossible to return back to the voting station and claim changing mind and voting again. Further, when the voting "day" is over, votes are counted in each voting station and this would generate the so called "raw" results, reflected in a list of candidates and corresponding numbers. Finally, all such lists are "brought together" as the source for calculating the overall results per country and / or per town and / or per county, and so on.

3.2 Assumptions

The above is the starting point for developing our conceptual model. For this reason, we should know *what to keep "in"* and *what to leave "out"*, and this decision is inevitably technology-driven because we leave out things that cannot be (at all) dependent on what the technology can offer. For example, if Steve is going to vote from home via computer and his father is standing behind him, influencing his vote, then we cannot do anything about this no matter how advanced technology is – this is just matter of each person's observing and upholding the rights each of us has. We thus make several assumptions: (i) issues of societal relevance which are nonetheless beyond the information system control are outside the scope of this research (consider the above example of Steve's voting). (ii) We assume sufficient IT literacy among the population, realizing nevertheless that there are poorly developed regions where this is not yet the case. (iii) We assume a democratic country in which governments do not manipulate the election process. The above assumptions maybe (partially) IT-inspired but even so, the assumptions have straightforward impact with regard to the technology-independent conceptual model because they lead us to what to leave out of the model.

3.3 The Voting Model

In order to keep (as promised) the model abstract (thus simple) but also properly focused, we: (i) apply only two concepts, namely `actor-role` and `relationship`; (ii) take into account what was presented in Sub-Section 3.1 and Sub-Section 3.2.

Our Actor-Role (AR) concept is consistent with Enterprise Ontology (see Section 2), suggesting that 'actor' is the (human) entity executing a particular task while 'actor-role' is about specifying the task itself, abstracting from who exactly is fulfilling this. For example, if a Professor is sending fax, then (s)he is doing something that is part of what the Secretary normally does and for this reason, determining the 'actor-role' here points to the label 'SECRETARY' (not 'PROFESSOR'). Our Relationship (R) concept is consistent with the SDBC approach (Shishkov, 2005) and is about whether or not collaboration is needed between two ARs that is necessary for one or both of them to deliver what they have to deliver.

Hence, we have identified the following ARs in the context of what was presented in this section:

AR1 – **CAMPAIGNER**: the one(s) campaigning in favour of a particular policy / party / vision and influencing the people in that way;

AR2 – **VOTER**: the one(s) voting for parliament / president / … and thus executing basic rights in the country;

AR3 – **PRIMARY COUNTER**: the one(s) counting the votes in a particular voting station;

AR4 – **SECONDARY COUNTER**: the one(s) aggregating the final result, by putting together the voting results from the voting stations;

AR5 – **ORGANIZER**: the one(s) organizing the voting process and supporting all above-mentioned accordingly;

AR6 – **CONTROLLER**: the one(s) controlling all above-mentioned;

7 – **SYSTEM**: even though this is not an actor-role, we have to somehow model abstractly the "place holder" where all voting "goes".

Further, we have identified the following Rs:

AR1-AR2 suggesting that the CAMPAIGNER is promoting political messages that are supposed to influence the VOTER;

AR2-SYSTEM suggesting that the VOTER provides essential input to the SYSTEM, namely the vote;

SYSTEM-AR3 suggesting that the SYSTEM has impact with regard to each voting station (said otherwise, each voting station has its "own" SYSTEM), by providing the information needed by the PRIMARY COUNTER for calculating the station results;

AR3-AR4 suggesting that the SECONDARY COUNTER needs the PRIMARY CONTER's feedback from each voting station, in order to aggregate the overall voting results;

AR5-ALL suggesting the ORGANIZER of the elections has relationship with all above-mentioned ARs and the SYSTEM as follows: creating conditions for the CAMPAIGNER to do promotion adequately; establishing that the rights of the VOTER are guaranteed; establishing rules and mechanisms according to which the PRIMARY COUNTER and the SECONDARY COUNTER should fulfil their corresponding tasks; establishing and running the voting SYSTEM;

A6-ALL suggesting that the CONTROLLER should execute effective control concerning all above-mentioned ARs and the SYSTEM, as guarantee that the voting is fair.

This is the basis for our conceptual model and as mentioned before, we abstract from several issues (as according to Sub-Section 3.2) and an example of this is that we do not consider an AR pointing to the one(s) (outside the CAMPAIGNER) who may be somehow influencing the decision of the VOTER – this could have been modeled as an AR by itself but we have not done this because of the lack of technical relevance, as explained above.

We present our conceptual model on Figure 1 and we use simple and intuitive graphical notations: the labels of the ARs are put inside boxes and the SYSTEM is presented as oval, while the Rs are represented as lines (the arrows indicate who is ADDRESSED in the relationship – for example: if the CAMPAIGNER is influencing the VOTER, then the arrow should be at the VOTER end because the VOTER is addressed by this).

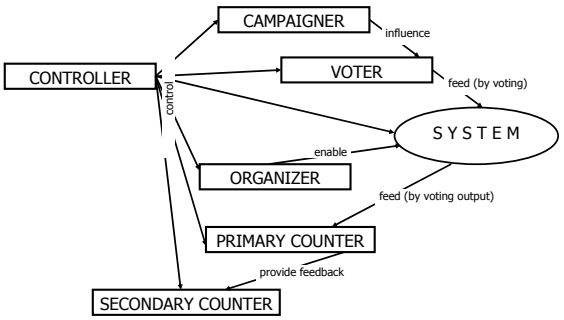


Figure 1: The Voting Conceptual Model.

As seen on the figure, we have not only drawn arrows at each line (lines representing Rs) but we

have also added labels there: the CAMPAIGNER would influence the VOTER, the ORGANIZER would enable the SYSTEM, and so on.

We claim that such an essential conceptual model is a good basis for further elaborations in different directions, such as structure (where we model entities and their relations), behaviour (where we model all activities in sequence, parallelism, and so on), data, and so on. The model would stay as "guarantee" for the inter-model consistency among structural, behavioural, data, and other "sub"-models. For the sake of brevity, we are not going in more details in presenting and discussing those issues. Still, the SDBC approach provides useful insight in this direction (Shishkov, 2005).

In line with the goals of the current paper, as mentioned in the Introduction, we go for extending the conceptual model, by identifying (technology-inspired) requirements, as part of the effort of supporting the development of eVoting systems. This will be addressed in the next section.

4 eVOTING REQUIREMENTS

In the current section, we will firstly elaborate the public demands (see Sub-Section 3.1) and then we will introduce our way of modeling requirements, being certainly restricted by the conceptual model (see Sub-Section 3.3).

4.1 Public Demands' Elaboration

In the current sub-section, we elaborate the previously listed public demands towards eVoting, taking into account that all those issues concern the people and the technology (what the current technological possibilities are), and the legislation. The current demands' elaboration would be useful as basis for our reflecting the demands in corresponding technical requirements.

With regard to the SECRECY OF VOTE demand, there are two things: (i) it is to be guaranteed that **nobody can know how a person has voted**; (ii) it is to be ensured that the person has been marked as "voted", such that (s)he **would not go to vote again**.

With regard to the COST ADEQUACY demand, the only way of avoiding the "big expensive black box" is to conceptualize the eVoting process such that **it is known what technology is needed for what**.

A way to guarantee against VIOLATIONS with regard to the way the System is working, is to present the user with **a simple and exhaustive list of options**, with no possibilities to do anything outside the presented options.

A way to guarantee agains MANIPULATIONS OF THE FINAL RESULTS is to keep things at two levels, such that the **Primary Counters generate the "raw" results** based on which the **Secondary Counters generate the final results** and this all stays stored with **possibility to check in the future**.

The COMPUTER-SERVER communication is to be such that there is guarantee that **a "packet" sent by a computer is received by the server and by noone else**; this is a matter of organization and also a matter of networking protocols.

CONTROLLABILITY can be partially achieved if **all intermediary results get transparent** and then the only remaining challenge is how are the "raw" results generated.

FAULT REACTION is a matter of **recoverability** and this is a non-fuctional concern that has to be addressed from a functional perspective nevertheless.

EASE OF USE is a matter of design.

The issue on QUALIFICATIONS needed for being involved in eVoting is a matter of legislation; as it was mentioned before, sufficient IT literacy among the population is assumed.

4.2 Way of Modeling Requirements

We consider OS norms (see Sub-Section 2.2) as helpful in the process of specifying requirements in this context not only because Organizational Semiotics is well-known for its strengths with regard to capturing societal aspects but also because OS norms have been researched also as useful with regard to requirements identification and specification. In particular, it has been studied how OS norms can help deriving use cases from business processes (Shishkov, 2005).

OS norms determine the conditions and constrains in controlling optional and conditional actions. They govern the behaviour of actors (agents), normally to decide when certain actions are performed. OS norms define clearly the roles, functions, responsibilities and authorities of the actors, for example:

`whenever` John is Customer of VISA
`if` VISA increase John's credit card limit
`then` John
`is` allowed
`to` use more credit.

4.3 The eVoting Requirements

Starting from unstructured information – the public demands presented in Sub-Section 3.1 (a mixture of technology-independent and technical issues), we had to firstly capture the functional gist (the conceptual model), abstracting from technical details – see Figure 1. Only on this basis, it was possible to adequately elaborate the public demands (see Sub-Section 4.1), making sure that each (technical) detail properly fits in the "big picture".

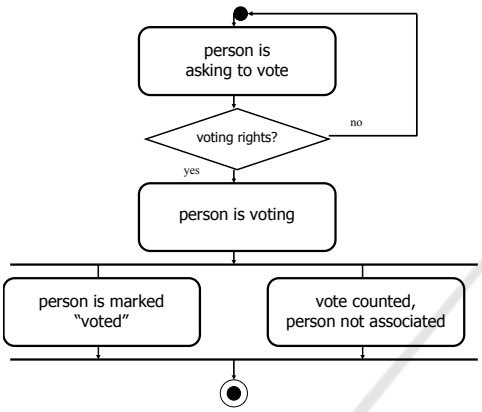


Figure 2: Workflow Pattern Corresponding to OS Norm 1.

Hence, inspired by Organizational Semiotics, we can now specify eVoting requirements, by means of OS norms, and in order to achieve better visualization and possible good basis for simulation (it this may be needed), we also reflect the OS norms in workflow, as studied by Shishkov (2005). Further, the demands being considered are many while the scope of the current paper is limited; hence, for the sake of brevity, we take only several of them (planning the rest as future work) and reflect them in the specification of requirements and in the next section, we briefly discuss the next step: "requirements-to-architecture".

We take in this section several eVoting public demands and we reflect them in specified requirements expressed in terms of OS norms. We start from the SECRECY OF VOTE one:

OS Norm 1:
`Whenever` John has voting rights
`if` John is executing eVoting
`then` the eVoting system
`is` (i) obliged `to` mark John as "voted"
`is` (ii) prohibited `from` recording the way John has voted.

Based on OS Norm 1, we derive a workflow pattern expressed with the notations of UML Activity Diagram (UML, 2016) – see Fig. 2.

With regard to the VIOLATION public demand, we formulate the following norm:

OS Norm 2:
`whenever` John is executing eVoting
`if` John is attempting a not allowed action
`then` the eVoting system
`is` prohibited `from` taking any action.

Based on OS Norm 2, we derive a workflow pattern – see Fig. 3.

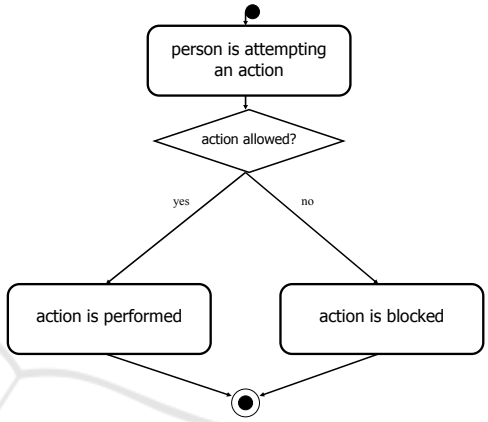


Figure 3: Workflow Pattern Corresponding to OS Norm 2.

5 TOWARDS A SERVICE-ORIENTED ARCHITECTURE

With regard to the "requirements-to-architecture" step, we consider the SDBC Approach (Shishkov, 2005) allowing for a component-based alignment between enterprise modeling and software design, ending up in specified software components that fit in the architecture. Those components we relate to web services in a way that what we see as web service "manifestation" is the functionality delivered by corresponding underlying software component(s). Further, we realize that in currently dealing with distributed cloud applications, it would often be that different software components have different origin thus not belonging to the same software application. Still, the SDBC Architecture is that needed abstraction which establishes and keeps the overall system "logic" no matter if a particular task is realized by the software application –to-be or by (dynamically) composed web services. Further detailing is left for future research. What we present in the current section is the SDBC-orientation we take with regard to architecture, noting the useful relation

to web services. For this reason, we briefly present below the SDBC Approach, SDBC standing for: '`Software Derived from Business Components`'.

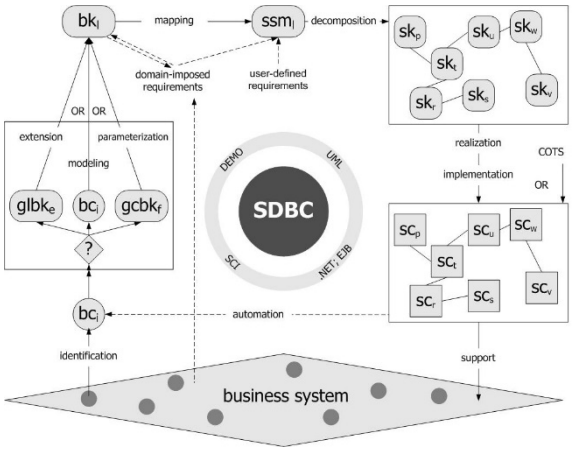
Firstly, SDBC assumes 4 modeling perspectives, namely: *Structural Perspective* that reflects entities and their relationships; *Dynamic Perspective* that reflects the overall business process and corresponding to this – the states of each entity, evolving accordingly; *Data Perspective* that reflects the information flows across entities and within the business process; *Language-Action Perspective* that reflects real-life human communication and expression of promises, commitments, etc. as also relevant to soundly building an exhaustive enterprise model.

Secondly, among SDBC's underlying theories are Enterprise Ontology and Organizational Semiotics (Shishkov et al., 2006) - see Section 2, which makes the approach siutable especially in the eVoting context of the current paper.

Thirdly, among the main SDBC concepts are the following:

- `Component` vs `CoMponent`: while components represent part of the whole, coMponents reflect a model of a component adequately elaborated in all 4 perspectives (see above), and we could thus have business components (business sub-systems) and software components (pieces of implemented software) as well as business coMponents and software coMponents, respectively;
- `General` vs `Generic`: those concepts are both about re-use, still – general is about re-using an abstract core (a general reservation engine, for example) while generic is about parameterizing something that is multi-specific (a car system to be adjusted to automatic or gear regime, for example);
- `Software Specification Model` – this is a technology-independent functionality model of the software system-to-be.

To summarize the SDBC outline, we use Fig. 4:



Abbreviations:

bc – Business Component
bk – Business CoMponent
glbk – General Business CoMponent
gcbk – Generic Business CoMponent
ssm – Software specification model
sc – Software Component
sk – Software CoMponent

Figure 4: SDBC – Outline (Shishkov, 2005).

As seen from the figure, we consider a Business System from which a Business Component(s) is to be identified and then reflected in a relevant model – a Business CoMponent. Another way for arriving at a Business CoMponent is by applying re-use: either extending a general Business CoMponent or parameterizing a generic one. Then, the Business CoMponent should be elaborated with the **`domain-imposed requirements`**, in order to add elicitation on the particular context in which its corresponding Business Component exists within the Business System. Then, a mapping towards a software specification model should take place and the **`user-defined requirements`** are to be considered, since the derived software model should reflect not only the original business features but also the particular requirements towards the software system-to-be. The software specification model in turn needs a precise elaboration so that it provides sufficient elicitation in terms of structure, dynamics, data and language-action –related aspects. It needs also to be decomposed into a number of Software CoMponents reflecting functionality pieces. Those CoMponents then are to undergo realization and implementation, being reflected in this way in a set of Software Components. Some Software Components could also be purchased. The Software Components are implemented using Software Component technologies, such as .NET or EJB, for instance. Finally, the (resulting) component-based ICT application would support informationally the target Business System, by automating anything that

concerns the considered Business Component (identified from the mentioned system).

SDBC was just briefly introduced above. Still, interested readers can find more information on SDBC in (Shishkov, 2005).

As mentioned in the previous section, we just briefly discuss our views on the "requirements-to-architecture" step in the eVoting context, and we leave it for future research to go in depth in this direction, possibly adapting some SDBC features accordingly, and of course validating our results by means of case studies. Still, it becomes clear how we proceed `from requirements to architecture` and how we consider `service-orientation`.

6 CONCLUSIONS

Understanding eVoting systems requires understanding the legislative and societal context. In this position paper we presented guidelines to come from global to detailed requirements, and then – to (service-oriented) architecture, based on Enterprise Ontology, Organizational Semiotics, and the SDBC Approach. The essence is that multiple theories need to be employed to understand and elicit requirements. Formal modeling should ensure that the requirements are consistent and meet the societal expectations.

REFERENCES

Dietz, J.L.G., 2006. *Enterprise Ontology, Theory and Methodology*. Springer-Verlag, Berlin Heidelberg.

Insfran, E., Pastor, O., Wieringa, R., 2002. Requirements Engineering-Based Conceptual Modelling. In *Requirements Engineering journal, Vol. 7, Nr. 2, 2002*. Springer-Verlag.

Konstantinov, M., Pelova, G., Boneva, J., 2009. Mathematics of the Bulgarian Electoral System. In *AMEE'09, 35th International Conference on Applications of Mathematics in Engineering and Economics*. AIP.

Liu, K., 2000. *Semiotics in Information Systems Engineering*. Cambridge University Press, Cambridge.

Papazoglou, M., 2012. *Web Services and SOA: Principles and Technology*, Prentice Hall, 2nd edition.

Plevneliev, R., 2016. *Statement by President Rosen Plevneliev at His 4th Annual Press Conference*, Published on the website of the President of the Republic of Bulgaria: http://www.president.bg.

Scammell, R., 2016. *Internet Voting a Success in Two European Countries*, Published on the European University Institute website: http://www.eui.eu.

Shishkov, B., 2005. *Software Specification Based on Re-usable Business Components (PhD Thesis)*, TU Delft – SIKS Publishing. Delft.

Shishkov, B., Dietz, J.L.G., Liu, K., 2006. Bridging the Language-Action Perspective and Organizational Semiotics in SDBC. In *ICEIS'06, 8th International Conference on Enterprise Information Systems*. SCITEPRESS.

UML, 2016, the website on the Unified Modeling Language: http://www.uml.org.

Van der Aalst, W., 2011. *Process Mining - Discovery, Conformance and Enhancement of Business Processes*. Springer-Verlag, Berlin Heidelberg.