A case for process-driven models for e-governance architectures

T B Dinesh
Janastu Bangalore, India
dinesh@janastu.org

Venkatesh Choppella
IIIT Hyderabad, India
venkatesh.choppella@iiit.ac.in

***Abstract*—Because of their potentially wide impact, e-governance systems beg the question of validation. How do we know an e-governance implementation *does* what it is supposed to do? How do we even know *what* it is supposed to do? Such questions are routine in the field of software engineering and are referred to, respectively as verification and specification. Verification and specification are tied together via a model. Software engineers call this model-driven design. The models most suitable for e-governance are a combination of data and processes. We introduce such a process-driven meta-model and show how it could usefully describe systems with e-governance behaviour.**

***Keywords*-E-governance, specification, validation, model-driven design, process, meta-model**

I. Introduction and Motivation

Critical and wide-scale impact of e-gov systems: E-governance software is defined as software implemented for governance. What distinguishes e-governance software from other software? It is its potential for critical impact on a large number of people. An incorrect or unpredictable behaviour in the functioning of this software therefore could have large, negative impact not only on citizens directly using this system, but other systems connected to it.

E-governance service-oriented and process-driven: Three aspects of E-governance applications are worth attention: First, their service-oriented nature. This naturally implies that the recipients of the services often include ordinary citizens, who are not passive, but are active agents of interaction with the governance via the application. This leads to the second and more technical feature of salience in E-governance applications. These applications should be understood as processes (in the computer science sense of the term). Third, e-governance processes manage data, which is often interfaced as documents and forms accessible via the web.

Impact of interfaces on data: Earlier forms of E-governance applications moved from documents on paper to documents in a database. These databases were accessible by database administrators. On the one hand, the web (and potentially mobile) has opened the opportunity for the public access of these documents, and is often the most used interface to the application. On the other hand, internal access and management, including mutations to these documents is also through the same web interface. Therefore, the footprints of mutation of records are also accessible via the web.

Impact of process-view on data: One important issue about documents and records is "who modifies what when?" Database-driven architectures handle this issue using access rights, granting permissions, etc. Databases are designed for documents, not processes. For example, users and their states *evolve* over time, which affects their access rights to the database. Unfortunately, traditional data models do not capture the dynamic nature of users, their states with respect to the workflow state of the processes that together define an application.

Objective of paper: The goal of this paper is to argue for a *model-based* approach to the design of e-governance systems to address the related problems of *specification* and *verification* of e-governance software. The paper begins with making the case for why the specification and verification assume importance in the realm of e-governance. We then present a meta-model of e-governance systems. Systems are modeled as processes, in the computer science sense. One advantage of this model is that it lets us think of families of systems with *e-governance behaviour*.

Approach of paper: The approach we adopt in this paper emphasises models of e-governance based on processes. The important issue of data security comes in as an issue concerning information components and individualized access rights to them, which *evolve* with the state of the process. Then, drawing on principles of software engineering, we introduce a meta-model for implementation of e-governance software. The meta-model defines the vocabulary and components needed for building e-governance models. Models are specified as abstract state machines. Interactions are modeled as transitions. The systems's data model is built as a collection of documents, fields and views.

Non-objectives of the paper: The goal of this paper is to argue for a model-based approach for e-governance applications and present a simple meta-modeling framework for specifying e-governance models. We do not discuss how e-governance implementations are to be designed or verified. Our focus is on the need for constructing models that embed the semantics of the application against which implementations may be verified. Furthermore, we do not

claim that the meta-model is complete in the sense that all e-governance models may be specified in this framework. Several other plausible alternatives for meta-models exist. We compare our approach with these other meta-modeling frameworks. But we make no attempt to relate our meta-modeling framework with a software modeling environment for constructing e-governance models and reasoning with them, nor do we present a methodology or technology for verifying the implementations of the models. These clearly belong to future work.

Paper roadmap: The rest of the paper is divided into the following sections: In Section II, we motivate the main ideas of the paper with a small example of an e-governance scenario in which an ombudsman attends to a citizen complaint. In Section III we present a meta-model for e-governance systems. In Section IV we specify an interacting process model the citizen complaint example and discuss the evolution of the citizen complaint workflow in terms of its security properties. In Section V, we briefly survey related efforts at model-driven approaches for e-governance. In Section VI, we conclude by highlighting future work.

II. Motivating example

Complaint reddressal by ombudsman: To help ground some of the ideas in this paper, we consider a small e-governance example: a citizen filing a complaint to a government ombudsman who reddresses the complaint. The citizen then provides feedback about the handling of the complaint. The example consists of two agents: the citizen and the ombudsmen interacting with each other and a document which evolves with the addition of fields (complaint, reddressal, feedback). The evolution of the interaction drives the evolution of the document. A further concern is security: once the citizen has submitted the complaint, is she allowed to modify it? Is anyone else viewing the document (let's say the database administrator) allowed to modify the complaint? Is someone else in the ombudsman's office allowed to make notes about the complaint? Thus an application implementing this example needs to ensure that specific parts of the document are *altered* in specific ways by specific users at specific times (and not before or after). Clearly, the constraints about who can access what when needs to be *specified* explicitly so that we may then *verify* whether the implementation is conforming to the specification. A typical e-governance application may consist of hundreds of such access constraints. The first step in designing an e-governance system is therefore to build a model of the system where such rules are made explicit.

III. A meta-model for e-governance systems

The e-governance framework (technically, the *meta-model*) we describe here provides a uniform vocabulary and a set of reasoning principles for specifying *domain-specific* models. The framework we describe is primitive, but with enough richness to capture simple, non-trivial models involving *processes* interacting through *events* and sharing data. Our metamodel is based on the process calculus of communicating concurrent systems (CCS) [1]. For example, we do not employ the full generality of *channels* available in the pi-calculus[2].

Processes and data: The framework model of e-governance we present consists of two primary components: processes and data. Each e-governance application is then modeled as a choreography of processes working to maintain some data.

States and events: Processes have *state* and interact with each other via *events*. Events could be external or internal to a process, or be the interaction between two processes. The occurrence of an event potentially changes the state of one or more of the process participating in the event.

Documents and fields: The primary unit of data in the metamodel is a *field*. A field has a name and a value. A document is a collection of fields. Fields have content. The content of the field may change, and this change may be effected by any agent over the process lifecycle of the agent. A single field may be modified by multiple agents at different times. (We ignore concurrent updates, whose handling is outside the scope and relevance of this paper). Fields have their own identities and may be shared across documents.

View evolution: To implement the notion of "who can see what when", we define the notion of *permissions* and *views*. A permission is either invisible (–), read-only (r-), write-only (-w), or read-write (rw). Each agent can view a subset of the documents in the system. For each document, the view consists of a set of fields. Furthermore, the view also determines the permission of the field. The key aspect of the meta-model is that *the view is a function of the agent's state*.

IV. Model for Ombudsman reddressal

Citizen and ombudsman processes: In this section, we build a model for the citizen-ombudsman interaction. The model consists of two agents, citizen and ombudsman. The diagram in Figure 1 shows the state evolution. The citizen agent is in one of the four states c_f (filing), c_w (waiting), c_e (evaluating), or c_d (done). The ombudsman agent is in one of the four states o_r (ready), o_h (handling), o_w (waiting), or o_d (done).

Interaction events: The two processes interact at event boundaries. Each event is in a send/receive pair. The three events in the system are complaint, response, evaluation. The citizen starts the "filing" while the ombudsman is in the ready state. The citizen then executes asynchronously a "send complaint" event and transitions to the "waiting" state. The ombudsman meanwhile, executes a "receive complaint" event, which takes the ombudsman to the "handling" state.

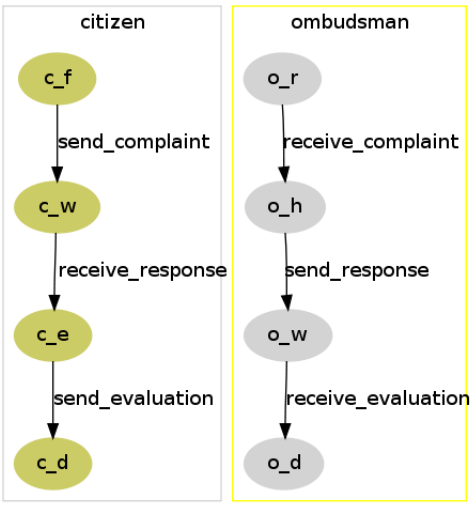

Figure 1. State machine for citizen complaint reddressal by Ombudsman

The other transitions are similar and self-explanatory. Note that we can view the system as one big process consisting of the citizen and ombudsman subprocesses. The states of the big process are a subset of the cross product of the subprocess states

$$\{(c_f, o_r), (c_w, o_h), (c_e, o_w), (c_d, o_d)\}$$

A. Data model evolution

Components of the data model: The pair of state machines specify the high-level control structure of the e-governance application. The next component of the complaint reddressal model is the specification of the data model. We assume that both the citizen and the ombudsman access a common data model, which *evolves* according to the combined states of the processes. The data model consists of the set of documents, a set of fields, and a view, which for a given system state, an agent, and the document, shows the set of fields and their permissions.

Initial state of the data model: The data model consists of one document called the "reddressal form." However, the evolution of the citizen reddressal process results in the addition of new fields to the data model. Initially, i.e, in the system state (c_f, o_r), there is one field *complaint*. In this state, the citizen's view of the *reddressal form* form consists of the *complaint* field, which is readable and writable. The ombudsman does not see the reddressal form or any fields in it. Fields that are readable and writable are marked in CAPITALS and colored olive green. Fields that are read-only are marked brick red.

View after send/receive complaint interaction: Once the citizen sends the complaint (eg., by submitting it via an online form, or by email, etc.), and the ombudsman receives it, each can view the reddressal form as consisting of the complaint field, which is now read-only. In addition, the ombudsman is handling the complaint by writing into the new, *answer* field. This field is invisible to the citizen and therefore not part of her view.

State	Citizen-view	Ombudsman-view
(c_f, o_r)	COMPLAINT	
(c_w, o_h)	complaint	complaint, ANSWER
(c_e, o_w)	complaint, answer, EVALUATION	complaint, answer
(c_d, o_d)	complaint, answer, evaluation	complaint, answer, evaluation

Figure 2. Data model evolution

View after send/receive answer interaction: Once the ombudsman answers the complaint, and the citizen receives it, each can continue to view the reddressal form's *complaint* and *answer* fields, but neither can edit them any longer. The citizen now begins to fill out an *evaluation* field in the reddressal form.

View after send/receive evaluation interaction: The citizen completes the evaluation and sends it. The ombudsman receives it. Both parties are now done. The fields *complaint*,*answer* and *evaluation* are now visible (but not editable) to both the citizen and the ombudsman.

Model formalizes access: There are several things to note about the above example. First, the model abstracts the control flow of the complaint reddressal process. The views formalize the constraint that access to the system's data (form and its fields) is governed by the state of the system, or correspondingly, the state of its agents. Note that the access is not governed just by who has access, but also *when*. (Access in one state does not guarantee access in a later state.)

Variants of model: It is easy to generalize the above model for variants of the problem: the ombudsman handling multiple complaints simultaneously, the ombudsman delegating the task of reddressal to others in the government office who get to add annotations and notes to the complaing, submitting multiple reddressal forms etc. The meta-model is capable of handling each of these variants.

V. Related approaches and research work

A vast number of e-governance systems today are data-centric, reflecting the need for e-governance applications to be front ends to data repositories. The main model for data-repositories are relational and object-oriented data bases. Relational databases have been standard technology for over two decades now. Object and object-relational models better capture the semantics of data in terms of operations (methods) embedded in objects as opposed to data records, which lack behaviour.

Two more recent approaches have been highly successful: SOA [3], [4], and process-workflow [5]. SOA, or *service oriented architectures* help define a system as a web service which interacts with other services over standard web-based (HTTP) protocols. The second approach, which is closer in nature to the work in this paper, is the business process-workflow approach. Business-process workflows are popular in Enterprise Resource Planning (ERP) systems. Standards like BPEL (Business process execution language) and BPMN (Business process modeling notation) are used to specify business processes and implement them on existing commercial enterprise servers. Semantic models for process workflows has been an important research problem in computer science. Petri-net models, a well-established formalism for studying concurrent systems, have been used to model e-governance systems [6]. Software Engineering approaches include the use of state charts [7]. More recently, approaches based on the pi-calculus [2] have been used to specify workflows [8]. Specialized programming languages like Orc provide a high-level language with block structure to specify workflow orchestrations over web [9]. However, all the workflow models cited here focus on control and process-flow aspects of the application. Integration with existing data models is not addressed. Such integration of the data model and the process model is necessary to address issues such as security and access rights.

E-governance systems in the small: The meta model presented here grew out of an implementation of a generic forms based, information management application for school governance [10] built on the Pantoto software [11], [12]. Various other contexts that use Pantoto show e-governance behaviour [13]. School information management is about lifting the paper/books based management of admissions, grades, parent-teacher meetings, library, et.al., while the existing processes and workflows are also lifted appropriately, to electronic/computer/internet medium. While a number of e-governance platforms are developed in the small and in the large by ethusistic proponents of e-governance, it begs to ask what is it about these software that make them specifically e-goverance software other than the fact that most of them deal with data of their citizens which other wise would be in the books in their offices. Of particular interest to us, is how secure and reliable these are compared to the traditional approach and how the software development process has been able to properly reflect and lift the traditional processes. A specific instance of this issue is indicated by the realistic concern of the principal of a school who was afraid of potential mutations by a system admin of student grades without her knowledge. Considering the school grades process as a formal process workflow model would help in two ways. First, the model could be treated as a specification that prohibits such a possibility by precisely outlining the states that are permitted to mutate the grades in question. Second, any actual mutation by a "third party" immediately manifests as a *violation* of the specification of the grades process.

VI. Future work and Conclusions

Future work: The meta-model we have presented here is simple but low-level. Higher-level abstractions and constructs are needed to make the meta model practical enough for specifying full-fledged, large-scale e-governance systems. We are currently investigating extending the simple meta-modeling language presented here to handle complex workflows integrated with views.

Model-based approach for e-governance: We have argued for a model-based approach to E-governance architectures in this paper. We have presented a simple process-based meta-model for modeling routine e-governance transactions. The meta-model emphasizes the primacy of processes, which in turn drive the view of the system's data. This may be seen as a contrasting alternative to the more data-centric approach to e-governance. We have, through an example, shown how models can be constructed to capture the control flow of an e-governance system, and the evolution of the data view according to that control flow. Admittedly, our model needs further development to handle different workflow patterns, needs to address interoperability, and needs to be integrated into a verification environment. However, our goal in this paper is to highlight the need for such a model-based approach, and show how the initial steps of such an approach might look.

Conclusion: Government organizations investing on e-governance applications should view e-governance applications in terms of behaviour *vis-à-vis* their conformance to a specification of the e-governance problem. Formal modeling is a useful way to approach this and governments should demand the models along with the implementations. (It's like demanding the blue prints along with the building.) Process-based models are one alternative in the design and modeling space that implementors should consider adopting. Modeling is necessary for building robust implementations accountable to the requirements of the e-governance problem.

References

[1] R. Milner, *A calculus of communicating systems*. Springer, 1982, iSBN:0387102353.

[2] ——, *Communicating and Mobile Systems: the π-calculus*. Cambridge University Press, 1999.

[3] B. Chakravarti and V. Varma, "An enterprise architecture framework for building service oriented e-governance portal," nov. 2008, pp. 1 –6.

[4] U. Marjit, R. Roy, S. Santra, and U. Biswas, "A semantic web service based approach to e-governance," aug. 2009, pp. 232 –237.

[5] H. Smith and P. Fingar, "Workflow is just a pi process," 2003. [Online]. Available: http://www.bpm3.com/picalculus

[6] W. Ge and R. Nan, "Modeling and research of the e-government system based on petri net," in *International Conference on Computer Science and Software Engineering (CSSE 2008)*. IEEE, 2008, pp. 1086–1089.

[7] D. Harel, "Statecharts: A visual formalism for complex systems," *Science of computer programming*, vol. 8, no. 3, pp. 231–274, 1987.

[8] F. Puhlmann, "On the suitability of the pi-calculus for business process management," in *Technologies for Business Information Systems*, W. Abramowicz and H. Mayr, Eds. Springer, 2007, pp. 51–62.

[9] W. R. Cook, S. Patwardhan, and J. Misra, "Workflow patterns in Orc," in *Coordination Models and Languages*, ser. Lecture Notes in Computer Science, P. Ciancarini and H. Wiklicky, Eds., vol. 4038. Springer, 2006, pp. 82–96.

[10] T. B. Dinesh and S. Uskudarli, "Community software applications," in *Home Informatics and Telematics: ICT for the Next Billion*, ser. IFIP International Federation for Information Processing, T. M. A. B. K. Venkatesh, A.. Gonsalves, Ed. Boston: Springer, 2007, vol. Volume 241, pp. 103–112.

[11] S. Uskudarli and T. Dinesh, "Pantoto: A participatory model for community information," in *Proceedings DyD'02: Development by Design 02*, 2002.

[12] S. Uskudarli and T. Dinesh., "Pantoto: A model for managing communities in the context of semantic web." in *ICSD: International Conference on Semantic Web & Digital Libraries*, February 2007.

[13] T. B. Dinesh and A. Bala, "Towards an appropriate software: Motivations and case studies," in *Governance of rural information and communication technologies : opportunities and challenges*, H. Misra, Ed. New Delhi : Academic Foundation, 2009.