State Based Access Control for Open E-governance

Ankur Goel
International Institute of Information Technology Hyderabad
India
ankur.goel@research.iiit.ac.in

Venkatesh Choppella
International Institute of Information Technology Hyderabad
India
venkatesh.choppella@iiit.ac.in

ABSTRACT

The Millenium Development goals emphasize the need for building "open, rule-based, predictable and non-discriminatory" e-governance systems. However, building such open systems remains a challenge: on the one hand, the systems are required to be open, whereas, on the other, there is the need to preserve and protect private and confidential information of potentially millions of users. This requires that e-governance systems carry clear specifications of how access to users' documents are managed throughout an e-governance application's workflow.

We describe a modular, fine-grained, state-based model that can form the basis for specifying access control in e-governance service delivery workflows. The model consists of three layers: a data store, a workflow layer, and an access control layer connecting the two. The data store consists of fields and forms. The workflow is specified as concurrent processes each representing either a citizen or a government actor. The access control layer specifies, for each user (process), a view of the data store as determined by that user's state in the workflow.

Such modular specifications can guide the implementation and the verification of e-governance applications. We illustrate our model with two representative examples from the e-governance domain and a web-based prototype implementation of one of them.

Categories and Subject Descriptors

D.2.1 [**Software Engineering**]: Requirements/ Specifications;
D.4.6 [**Operating Systems**]: Security and Protection—*Access Controls*;
H.4.1 [**Information Systems Applications**]: Office Automation—*Workflow management;*
J.1 [**Computer Applications**]: Administrative Data Processing—*Government*

Permission to make digital or hard copies of all or part of this work for personal or classroom use is granted without fee provided that copies are not made or distributed for profit or commercial advantage and that copies bear this notice and the full citation on the first page. To copy otherwise, to republish, to post on servers or to redistribute to lists, requires prior specific permission and/or a fee.
ICEGOV 2013 Seoul, Republic of Korea
Copyright 2013 ACM 0-12345-67-8/90/01 ...$15.00.

General Terms

Security, Standardization

Keywords

Workflows, Access Control, Open Government

1. INTRODUCTION

The right of the citizen to scrutinize and participate in the process of governing forms the basis for Open Government [32]. The origins of the concept of Open Government date back to the era of European Enlightenment [21], whereas the term was formally introduced in the United States of America [32]. In the past decade, thanks to the initiatives by various governments, Open Government is being adopted across the globe. However, most of the effort till now has been towards publishing static data.

This basic idea of publishing government data and proceedings however continues to evolve, under the influence of the open source movement [24]. Open government at its current state means a government where citizens can not only access data and proceedings, but also contribute back by becoming participants. The three modern principles of open government are: Transparency, Participation and Collaboration [24]. Transparency forms the basis for open government. It is the principle by which citizens have access to government processes and actions. This principle is a part of the law in most countries of the world. For example, in India, the Right to Information Act (RTI) [20] gives the public the right to inspect government documents and records. The principle of participation on the other hand, allows the citizen to provide input and feedback to a government process. Collaboration is the principle where a citizen can actively take part in the government process. Collaboration ranges from the public being involved in the law making process to citizen-run community service efforts.

The United Nation's Millennium Development Goals (MDGs) [29] also recognize the need for open government. Goal 8 expresses the need for developing "open, rule-based, predictable and non-discriminatory" systems. In recent years, many governments in the world have moved towards standardization of e-governance processes to align themselves with the principles of open government [28, 31]. However, in order to make this vision a reality, governments must ensure that specifications of their process workflows are published and available for scrutiny.

These specifications can then form the basis for verifiable and auditable implementations. The problem here is close to that fou-

nd in software engineering requirement specifications, where formal notation is used to describe the required behaviour of an application with its users and the operating environment. These specifications are used to build implementations as well as validate them. In this paper, we approach the problem of specifying open e-governance applications by modelling them as service delivery workflows. We consider a government representative or a citizen as a user in the e-governance workflow. The principles of Open Government may then be translated as specific requirements for the workflow.

Transparency for a user in a workflow is simply read access to certain fields or documents in an e-governance workflow. Participation is a scenario where the citizen has access to certain documents such as feedback. Collaboration is similar to participation, however the citizen would act as an active user in the workflow, and thus can contribute to the process. Finally, the specification of the workflow should be made public to enhance the level of transparency even more.

Open government encourages the citizens to get involved in the e-governance process, however this also raises the issue of access control due to the sensitivity of data involved. The users should have access to view and edit certain data at only specific points of time, which should be dictated by the e-governance process on that data. Thus, the rights of an user to access data should depend on the state of the user with respect to the workflow. This can help address severe problems such as corruption. For example, in a tax collection process, a tax collector should never have access to edit the amount of tax to be paid by the citizen at any step in the workflow.

There have been many frameworks for modelling e-governance processes [6, 30, 33, 37], however most of them do not address the problem of access control. In the area of web systems, there are several approaches for state based access control, but they present no theoretical model, e.g. Websphere [1] and Drupal [2]. This approach works for simple workflows, but can lead to state explosion for large, complex with several users, because the access is defined on the state of the workflow rather than the state of a user. Another approach is Privilege State based Access Control(PSAC) [23], which assigns state to the data for each user/role. We, on the other hand, use a novel approach of assigning states to each process, where each process represents a citizen or a government entity in the workflow. This state is then used to determine the access of the user and also the government. Briefly in our model, the access of a user is based on his/her progress in the workflow.

This paper proposes a layered model of access control so that the principles of open government may be realized. In our model, the system state evolves by user interaction. Also, for every user, access to shared data in the workflow changes with every interaction. We employ the formalism of CCS (Calculus of communicating processes) [25] to specify the workflows layer. CCS is an algebraic specification language and allows workflows to be specified in a succinct way. CCS, being a formal language also allows for static and dynamic analysis of such workflows [13, 16, 43]. The data layer in the model is stored in form of fields and forms. The view layer connects the data and workflow by defining permissions on the data for each user state. The view layer is responsible for access control.

In this section we gave a brief introduction to open government, workflows and our model. In Section 2, we discuss the model and also give a brief introduction to CCS. In Section 3, we show via examples that how this model can be applied to e-governance applications. Section 4 discusses how our model can be implemented as a web application. Section 5 discusses related work. Section 6 concludes the paper with pointers to future work.

2. THE MODEL

The state based access control model consists of three layers: a data layer, a process layer to express workflows, and the view layer. The data layer is the representation of the data in the database. The process layer is the workflow management system which uses CCS to specify workflows. The view acts as a layer between the data and process model. This layer also addresses the issue of access control. It maps the user state in the workflow to the data model.

2.1 Data Layer

The model assumes a forms-based document store, thus the data layer has forms, fields and users. Forms are a collection of fields. However, fields are not statically assigned to forms by the data layer. Each field has content associated with it, but field content is not relevant to access control and is thus not part of this model. The basic types of the data layer are:

User : *TYPE*
Form : *TYPE*
Field : *TYPE*

The data model operates on a fixed set of users, forms and fields.

users : *set*[*User*]
forms : *set*[*Form*]
fields : *set*[*Field*]

2.2 Process Layer

We use workflows in order to define the process layer. Every user in the workflow is considered a separate process and has its own corresponding state with respect to the workflow. The set of actions available to the user is dependent on that user's state. For the specification of such workflows, we employ CCS, which is introduced below.

2.2.1 Introduction to CCS

We use the Calculus of Communicating Processes (CCS) to specify workflows for our model. Figure 1 captures the syntax of CCS. Readers already familiar with CCS may skip this introduction.

An action is either silent τ, or a send, or a receive on a channel a. The concurrent composition of processes P and Q is denoted $P|Q$. $\Sigma_i \pi_i.P_i$ denotes the process which upon interacting according to π_i becomes the process P_i. The operators | and + are assumed to be commutative and associative. However, actions do not distribute over summations. Thus $a.(P + Q)$ and $a.P + a.Q$ behave differently.

The dynamics of processes is defined by the basic reduction rules shown in Figure 2. The first rule, called reaction, allows a process expression with two concurrent processes $a.P + M$ and $\bar{a}.Q + N$, one with a send capability and the other with a receive capability to evolve into another pair of concurrent processes

$P|Q$.

Actions	π	:=	
		a	Receive
		$\mid \overline{a}$	Send
		$\mid \tau$	Silent action
Processes	P	:=	
		0	Empty process
		$\mid \Sigma_i \pi_i.P_i$	Guarded Summation
		$\mid P\|Q$	Parallel composition
Definitions	D	:=	
		$A \stackrel{\text{def}}{=} P$	

Figure 1: Syntax of CCS.

Note that the alternatives *M* and *N* are discarded. The second rule called "silent action" corresponds to the case when a process autonomously evolves into another process.

$$(\overline{a}.P + M)|(a.Q + N) \rightarrow P|Q$$
$$\tau.P + N \rightarrow P$$

Figure 2: Dynamics of CCS rules defined via reduction.

2.2.2 Workflows as interacting processes

We express workflows as a sequence of CCS process definitions, followed by a process expression which denotes the initial state of the workflow. A change in state for a user is called a transition. For every user we list the transitions, this forms the process definition of that user. The next section shows how this is done with the help of examples.

We use this formal specification, as it allows for the simulation of the workflow and can form the basis for a requirement specification for any vendor interested in implementing the workflow. Also, this specification can be analysed using various verification and validation tools available.

2.3 View Layer

The view layer acts as an interface between the data layer, the process layer and the users. The view layer uses permissions to control the access of data of all users based on their states. These permissions are not static in nature, but are determined by the state of the user in the workflow. This dynamic nature of permissions forms the core of our model. We consider a user to have one of these four permissions to a field: neither read nor write (−−), read only (r−), write only (−w), read and write (rw), the four combinations of read and write.

Perm : TYPE = {−−, r−, −w, rw}

Another function of the view is to define the form structure dynamically. In other words, the view determines what fields a form contains for each user.

The view can be thought of as every user's access to the system at any given workflow-state, which consists of a set of forms, each with a set of fields and their corresponding permissions. Formally, we define the view as a function that maps a user's state and a form to a set of fields and their permissions. We formalise this by defining *field-view* to be a partial map from fields to *Perm*. *Field-view* is defined for a user, and maps a subset of the system's fields to their corresponding permissions. *Form-view* is a partial map from forms to *field-view*. Form views provide a view to a subset of the system's forms. The user-view of a user is a total map from the user's state in the workflow to form-view.

field-view : *fields* ⇀ *Perm*
form-view : *forms* ⇀ *field-view*
user-view(u) : *States*(u) → *form-view*

A user's view, therefore, maps each state of the user to a subset of forms, where each form is mapped to a subset of fields which are in turn mapped to permissions. Note that views are not static; they are tied to the user's state in the workflow.

On a change of state a user, the view is completely rebuilt for that user. However, instead of defining the entire user's view for each and every user state, the view may be incrementally updated. An update changes the system's view by utilizing the view prior to the transition, and applying the update associated with the transition.

3. EXAMPLES

In this section we discuss our approach with two government service delivery workflows. The National Knowledge Commission of India reviews various e-governance efforts and recommends appropriate reforms for e-governance [28]. The primary recommendation is for all government processes to be re-engineered to suit computerization starting with 10-20 important processes and services. Our first example is one such process, the passport application in India, a workflow which demands a high level of transparency for the citizen, along with privacy for the user data. We take another example of a public grievance portal, a workflow which involves all the three principles of open government: transparency, participation and collaboration.

3.1 Passport Application process

Figure 3 presents a considerably simplified version of the Passport application process in India [19]. There are several fields in the form to be submitted to apply for the passport, however for the sake of simplicity we consider only three fields: Name, Date Of Birth(DOB), and Address. Each of the three layers of the model are described next.

3.1.1 Process Layer

Figure 4 shows the CCS process specification for this workflow. This forms the process layer of our model. There are three users in the workflow: Citizen(*c*), Passport Office(*ppo*) and the police(*pol*). A citizen in the workflow can be in one of the following states: filling the form *c-filling*, waiting for the response *c-waiting* and done with the process *c-done*. Similarly, the passport officer can be in one of the following states: waiting for the application from the citizen *ppo-waiting*, reviewing the application *ppo-reviewing*, waiting for police verification *ppo-verifying* and being done *ppo-done*. The police is in the following

1. The applicant (citizen) submits a form with personal details to the Passport Officer (PPO).
2. PPO receives the form. Then,
   (a) If the form is incomplete, the PPO sends back the form to the applicant
   (b) If form is complete, PPO forwards the application to the police department for verification.
3. Police receives the application form, and tries to verify the address of the applicant.
   (a) If the citizen's address cannot be verified, police sends "reject" to the PPO.
   (b) Else if the address is verified, police sends "approve" to the PPO.
4. Based on the response from the police, the PPO delivers the passport to the citizen.

Figure 3: Passport application: Plain text specification

states: ready for verification *pol-ready*, verifying the application *pol-verifying* and being done with processing *pol-done*.
This layer maps the entire control flow of the workflow. Figure 5 maps the plain text specification to the CCS based workflow specification.

$$
\begin{array}{lll}
1. & \textit{c-filling} \stackrel{\text{def}}{=} & \overline{\textit{submit}}.\ \textit{c-waiting} \\
2. & \textit{c-waiting} \stackrel{\text{def}}{=} & \textit{incomplete}.\ \textit{c-filling} \\
3. & & \quad + \textit{reject}.\ \textit{c-done} \\
4. & & \quad + \textit{approved}.\ \textit{c-done} \\
5. & \textit{ppo-waiting} \stackrel{\text{def}}{=} & \textit{submit}.\ \textit{ppo-reviewing} \\
6. & \textit{ppo-reviewing} \stackrel{\text{def}}{=} & \overline{\textit{incomplete}}.\ \textit{ppo-waiting} \\
7. & & \quad + \overline{\textit{verify}}.\ \textit{ppo-verifying} \\
8. & \textit{ppo-verifying} \stackrel{\text{def}}{=} & \textit{confirm}.\ \overline{\textit{approved}}.\ \textit{ppo-done} \\
9. & & \quad + \textit{fail}.\ \overline{\textit{reject}}.\ \textit{ppo-done} \\
10. & \textit{pol-ready} \stackrel{\text{def}}{=} & \textit{verify}.\ \textit{pol-verifying} \\
11. & \textit{pol-verifying} \stackrel{\text{def}}{=} & \overline{\textit{confirm}}.\ \textit{pol-done} \\
12. & & \quad + \overline{\textit{fail}}.\ \textit{pol-done}
\end{array}
$$

Initial Process Expression:

$$\textit{c-filling} \mid \textit{ppo-waiting} \mid \textit{pol-ready}$$

Figure 4: Passport application workflow

This model is defined in a way such that an action from a user is required whenever the user needs to send on a channel. However, receiving on the channel does not require any action by the user. For instance, a citizen has to only execute one action, which is to submit the application. This is shown in Step 2 of Figure 3. The PPO has to get the application verified by the police and return the result to the citizen. Note that the PPO can also send an incomplete application back to the user as many times as required. Similarly the police has just one action which is to confirm the application to the PPO, shown in Step 11-12 of the plain text specification shown in Figure 3.

Plain-text	Workflow	Users involved
Step 1	1,5	*c,ppo*
Step 2(a)	2,6	*c,ppo*
Step 2(b)	7,10	*ppo,pol*
Step 3(a)	3,9,12	*c,ppo,pol*
Step 3(b)	4,8,11	*c,ppo,pol*

Figure 5: Passport Application - Mapping of the plain text specification to the workflow specification

3.1.2 Data Layer

The data model for this layer is given in Figure 6. It stores the users, forms, fields as well as user-states. For this example, it can be seen that there is a single form and four fields. Note that fields are not mapped to forms by the data model. We shall discuss how this is done later in this section. It can also be seen that though the states have been incorporated in the data model, the data layer is disjoint from the process layer.

$$
\begin{array}{lcll}
\textit{users} & = & \{c, ppo, pol\} & \\
\textit{forms} & = & \{f\} & \text{Form} \\
\textit{fields} & = & \{\textit{name}, & \text{Name} \\
 & & \textit{dob}, & \text{Date of Birth} \\
 & & \textit{add}, & \text{Address} \\
 & & \textit{qstatus}\} & \text{Query status}
\end{array}
$$

Figure 6: Passport Application - Data Layer

3.1.3 View layer

The view connects the data and the workflow layers. The view layer associates user states with the form view. Figure 7 shows

$$
\begin{array}{lcllll}
\textit{c-filling} & \mapsto & \{f & \mapsto & \{\textit{name} & \mapsto rw, \\
 & & & & \textit{dob} & \mapsto rw, \\
 & & & & \textit{add} & \mapsto rw\ \}\} \\
\textit{c-waiting} & \mapsto & \{f & \mapsto & \{\textit{name} & \mapsto r-, \\
 & & & & \textit{dob} & \mapsto r-, \\
 & & & & \textit{add} & \mapsto r-, \\
 & & & & \textit{qstatus} & \mapsto r-\ \}\} \\
\textit{c-done} & \mapsto & \{f & \mapsto & \{\textit{name} & \mapsto r-, \\
 & & & & \textit{DOB} & \mapsto r-, \\
 & & & & \textit{add} & \mapsto r-, \\
 & & & & \textit{qstatus} & \mapsto r-\ \}\}
\end{array}
$$

Figure 7: Passport Application - Citizen's view

the view for a citizen corresponding to the citizen's state. Note how the data layer i.e., forms and fields are mapped to the user states with their associated permissions. This change in permissions with states is how the data-flow is expressed by our model.

Figures 8 and 9 are the views for the PPO and the police

$$
\begin{array}{lllll}
\textit{ppo-waiting} & \mapsto & \{f & \mapsto & \{\} \} \\
\textit{ppo-reviewing} & \mapsto & \{f & \mapsto & \{\textit{name} \mapsto r-, \\
 & & & & \quad \textit{DOB} \mapsto r-, \\
 & & & & \quad \textit{add} \mapsto r- \}\} \\
\textit{ppo-verifying} & \mapsto & \{f & \mapsto & \{\textit{name} \mapsto r-, \\
 & & & & \quad \textit{DOB} \mapsto r-, \\
 & & & & \quad \textit{add} \mapsto r-, \\
 & & & & \quad \textit{qstatus} \mapsto r- \}\} \\
\textit{ppo-done} & \mapsto & \{f & \mapsto & \{\textit{name} \mapsto r-, \\
 & & & & \quad \textit{DOB} \mapsto r-, \\
 & & & & \quad \textit{add} \mapsto r-, \\
 & & & & \quad \textit{qstatus} \mapsto r- \}\}
\end{array}
$$

Figure 8: Passport Application - PPO's view

respectively. It is important to observe that fields are not duplicated for each user. Instead the view to that field is dependent on the user's access to that field, which is a function of the user state.

$$
\begin{array}{lllll}
\textit{pol-ready} & \mapsto & \{f & \mapsto & \{\} \} \\
\textit{pol-verifying} & \mapsto & \{f & \mapsto & \{\textit{name} \mapsto r-, \\
 & & & & \quad \textit{DOB} \mapsto r-, \\
 & & & & \quad \textit{add} \mapsto r-, \\
 & & & & \quad \textit{qstatus} \mapsto rw \}\} \\
\textit{pol-done} & \mapsto & \{f & \mapsto & \{\textit{name} \mapsto r-, \\
 & & & & \quad \textit{DOB} \mapsto r-, \\
 & & & & \quad \textit{add} \mapsto r-, \\
 & & & & \quad \textit{qstatus} \mapsto r- \}\}
\end{array}
$$

Figure 9: Passport Application - Police's view

3.2 Public Grievance portal

The public grievance portal of India [18] is where any citizen can file a complaint or raise a grievance with any department of the government. There are three users involved in this process, the citizen who files the grievance, the Public Grievance officer (PGO) who is responsible for the grievance, and the government department against which the grievance is raised. In this process the identity of the citizen is hidden from the concerned government department. Figure 10 gives the specification of this process in plain text.

3.2.1 Process Layer

Figure 11 shows the CCS process specification for this workflow. There are three users in the workflow as indicated earlier: citizen(*c*), Public Grievance Officer(*pgo*) and the concerned government department(*gov*). A citizen can be in one of the following states: ready to file a grievance *c-ready*, waiting for the response *c-waiting* and done with the process *c-done*. Similarly, the *pgo* can be in one of the following states: ready to accept grievances *pgo-ready*, reviewing the grievance *pgo-reviewing*, waiting for the reply from the government department *pgo-waiting*, evaluating the response *pgo-evaluating*, or finally done *pgo-done*. *gov* can be in one of the following states: ready for processing the grievance *gov-ready*, addressing the grievance *gov-addressing*, waiting for the *pgo*, *gov-waiting* and done *gov-done*.

1. Citizen fills in the grievance form along with personal details. (Name, Address) and submits it to the PGO.
2. PGO receives the grievance and reviews it. Then the PGO either:
   (a) Rejects the grievance. Process ends here.
   (b) Or forwards it on to the concerned government department.
3. The concerned government department addresses the issue and responds back to the PGO.
4. The PGO evaluates the response. Then,
   (a) If the response is unsatisfactory, PGO rejects the response and waits for a new reply from the government department.
   (b) Else, the PGO accepts the response and also forwards it to the citizen.

Figure 10: Grievance Portal - Plain text specification

$$
\begin{array}{llll}
1. & \textit{c-ready} & \stackrel{\text{def}}{=} & \overline{\textit{submit}}\ .\ \textit{c-waiting} \\
2. & \textit{c-waiting} & \stackrel{\text{def}}{=} & \textit{respond.}\ \textit{c-done} \\
3. & & & +\ \textit{decline.}\ \textit{c-done} \\
 & & & \\
4. & \textit{pgo-ready} & \stackrel{\text{def}}{=} & \textit{submit.}\ \textit{pgo-reviewing} \\
5. & \textit{pgo-reviewing} & \stackrel{\text{def}}{=} & \overline{\textit{decline}}.\ \textit{pgo-done} \\
6. & & & +\ \overline{\textit{req}}.\ \textit{pgo-waiting} \\
7. & \textit{pgo-waiting} & \stackrel{\text{def}}{=} & \textit{reply.pgo-evaluating} \\
8. & \textit{pgo-evaluating} & \stackrel{\text{def}}{=} & \overline{\textit{accept}}.\ \overline{\textit{respond}}.\ \textit{pgo-done} \\
9. & & & +\ \overline{\textit{reject}}.\ \textit{pgo-waiting} \\
 & & & \\
10. & \textit{gov-ready} & \stackrel{\text{def}}{=} & \textit{req.}\ \textit{gov-addressing} \\
11. & \textit{gov-addressing} & \stackrel{\text{def}}{=} & \overline{\textit{reply}}.\ \textit{gov-waiting} \\
12. & \textit{gov-waiting} & \stackrel{\text{def}}{=} & \textit{accept.}\ \textit{gov-done} \\
13. & & & +\ \textit{reject.}\ \textit{gov-addressing}
\end{array}
$$

Initial Process Expression:

$$\textit{c-ready} \mid \textit{pgo-ready} \mid \textit{gov-ready}$$

Figure 11: Public Grievance Portal workflow

Figure 12 shows the map from the workflow specification map to the plain text specification.

Plain-text	Workflow spec	Users involved
Step 1	1,4	*c,pgo*
Step 2(a)	3,5	*c,gov*
Step 2(b)	6,10	*gov,pgo*
Step 3	7,11	*pgo,gov*
Step 4(a)	9,13	*pgo,gov*
Step 4(b)	2,8,12	*c,pgo, gov*

Figure 12: Grievance Portal - Mapping of the plain text specification to the workflow specification

3.2.2 Data Layer

The data model can be seen in Figure 13. There are four fields: name, address, grievance and response. There are two forms: grievance form and status form.

$$
\begin{array}{lcll}
users & = & \{c, pgo, gov\} & \\
forms & = & \{g, & \text{Grievance Form} \\
 & & r\} & \text{Response Form} \\
fields & = & \{name, & \text{Name} \\
 & & add, & \text{Address} \\
 & & grievance, & \text{Citizen grievance} \\
 & & response\} & \text{Govt response}
\end{array}
$$

Figure 13: Grievance Portal - Data Layer

3.2.3 View layer

Now we define the view layer for this example. The view associates user states with form view. Figures 14-16 capture the state-based access control for the citizen, PGO and the government department respectively.

$$
\begin{array}{lllllll}
c\text{-}ready & \mapsto & \{g & \mapsto & \{name & \mapsto & rw, \\
 & & & & add & \mapsto & rw, \\
 & & & & grievance & \mapsto & rw\ \}, \\
 & & \{r & \mapsto & \{\}\} & & \\
c\text{-}waiting & \mapsto & \{g & \mapsto & \{name & \mapsto & r-, \\
 & & & & add & \mapsto & r-, \\
 & & & & grievance & \mapsto & r-\ \}, \\
 & & \{r & \mapsto & \{grievance & \mapsto & r-\ \}\} \\
c\text{-}done & \mapsto & \{g & \mapsto & \{name & \mapsto & r-, \\
 & & & & add & \mapsto & r-, \\
 & & & & grievance & \mapsto & r-\ \}, \\
 & & \{r & \mapsto & \{grievance & \mapsto & r-, \\
 & & & & response & \mapsto & r-\ \}\}
\end{array}
$$

Figure 14: Grievance Portal - Citizen's view

The citizen uses the grievance form to submit a grievance, while the status form is used to collect the response from *gov* and show it to the citizen. It can be seen that after the process is complete, that the *pgo* views all four fields in the *grievance* form. On the other hand the citizen views the same fields, however in two different forms. This is modelled using the dynamic nature of the view.

There are more key points the may come to the observation of the reader in this example. For example the view in this model has been designed in a way such that the government department against which the grievance has been posted will not know the identity of the citizen who posted the grievance. This can be seen in Figure 16, where the view of the government department is defined such that the department does not have access to the

$$
\begin{array}{lllllll}
pgo\text{-}ready & \mapsto & \{g & \mapsto & \{\}\} & & \\
pgo\text{-}reviewing & \mapsto & \{g & \mapsto & \{name & \mapsto & r-, \\
 & & & & add & \mapsto & r-, \\
 & & & & grievance & \mapsto & r-\ \}\} \\
pgo\text{-}waiting & \mapsto & \{g & \mapsto & \{name & \mapsto & r-, \\
 & & & & add & \mapsto & r-, \\
 & & & & grievance & \mapsto & r-\ \}\} \\
pgo\text{-}evaluating & \mapsto & \{g & \mapsto & \{name & \mapsto & r-, \\
 & & & & add & \mapsto & r-, \\
 & & & & grievance & \mapsto & r-, \\
 & & & & response & \mapsto & r-\ \}\} \\
pgo\text{-}done & \mapsto & \{g & \mapsto & \{name & \mapsto & r-, \\
 & & & & add & \mapsto & r-, \\
 & & & & grievance & \mapsto & r-, \\
 & & & & response & \mapsto & r-\ \}\}
\end{array}
$$

Figure 15: Grievance Portal - PGO's view

grievance form. Note that this access of forms and fields is independent of the data model, but is controlled completely by the view.

The permission on response for the government department alternates between read-write and read-only for the *gov-addressing* and *gov-waiting* states respectively. This is a good example to show how the state impacts the view of a user. Another observation is that the field grievance is shared between both the forms. This is made possible by the delinking of forms and fields. This behavior can be considered similar to two different fields linking to the same content.

$$
\begin{array}{lllllll}
gov\text{-}ready & \mapsto & \{r & \mapsto & \{\}\} & & \\
gov\text{-}addressing & \mapsto & \{r & \mapsto & \{grievance & \mapsto & r-, \\
 & & & & response & \mapsto & rw\ \}\} \\
gov\text{-}waiting & \mapsto & \{r & \mapsto & \{grievance & \mapsto & r-, \\
 & & & & response & \mapsto & r-\ \}\} \\
gov\text{-}done & \mapsto & \{r & \mapsto & \{grievance & \mapsto & r-, \\
 & & & & response & \mapsto & r-\ \}\}
\end{array}
$$

Figure 16: Grievance Portal - Government's view

4. IMPLEMENTATION AS A WEB BASED APPLICATION

This section discusses a prototype web-based implementation of the model. This is available on GitHub [4]. The user-view function of the view is used to determine the list of forms, their fields and their corresponding permissions. Depending on the permission of the field, the field is dis- played to the user as readable, writable or both. A readable field is represented as a label with its value as the content, a writable field as an empty input box, and a read-write field as an input box pre-filled with the content. Note that the view only determines the structure of the page, the content of these fields is taken from the data layer. The buttons available to the user are determined by the process layer based on the transitions available from the current user

state. Each button click instantiates an action, based on which, the corresponding transition of state takes place.

We use the passport application example discussed in Section 3.1 to introduce the implementation. Figure 17 shows screenshots of the web application as viewed by the PPO.

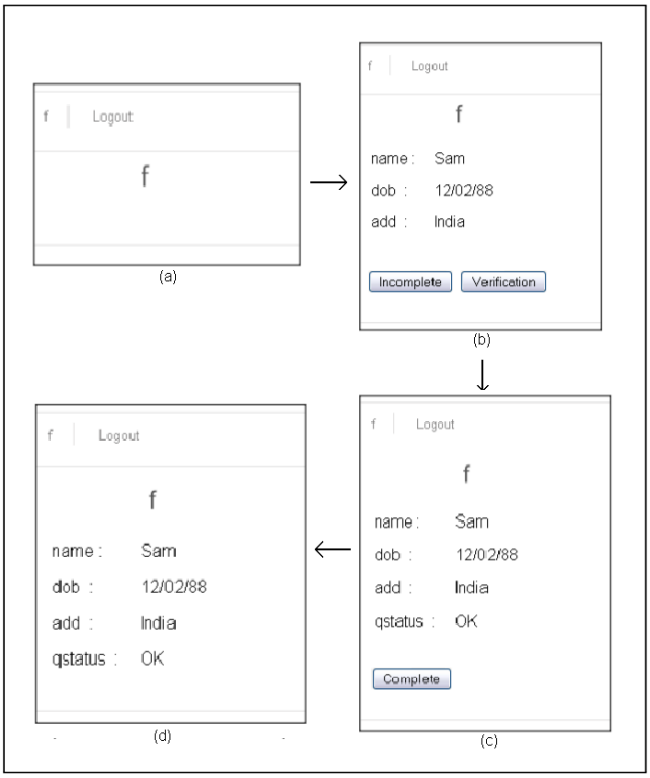

Figure 17: Screen shots of the passport application process as viewed by the PPO in various states: (a) ppo-waiting, (b) ppo-reviewing, (c) ppo-verifying, (d) ppo-done.

Figure 17(a) shows the *ppo-waiting* state, when the PPO is waiting for the citizen's application. This is an empty form as the PPO does not have access to any fields at that state. Figure 17(b) shows the *ppo-reviewing* state. In this state the PPO has read-only access to the *name*, *dob* and *add* fields, and can either send the application back by clicking "*Incomplete*" or send it to the police by clicking "*Verification*". Similarly, Figure 17(c) and (d) show when the verification is back from the police, and when the process is done.

5. RELATED WORK

In the past decade there have been several proposals that adopt the approach of workflows in e-governance. Punia [33] has proposed one such model for inter-organizational e-governance workflows. Punia lists various workflow engines that could be used, however none of them give the required access control required in such sensitive processes. Vergindas et al. [37] introduce a light modelling framework for e-governance service workflows focused on re-using recurring segments, however this framework doesn't provide a concrete model to implement such workflows, nor does it discuss the issue of access control. Beer et al. [6] have pro- posed an architecture for distributed workflow systems for e-government. They use the Wf-XML protocol, but again do not investigate access control aspect. Ruth et al. [11] propose an approach for model driven security for workflows in e-government. They use UML to specify security requirements. In India, Wf-Bhulekh [27] is a workflow based application for land record management. It uses OpenWFE as the workflow management system.

The remainder of the section focuses on the state of the art work going on in the fields of workflows and access control outside the domain of e-governance. We will first talk about other models used to express workflows and business processes.

Business Process Model and Notation (BPMN)[39, 40] is a notation for specifying business processes in terms of a Business process diagram (BPD). It is widely used in the industry to model business processes. The specifications are formulated as diagrams and graphs. Business Process Execution Language (BPEL) [14] is a process specification language which allows for specifying interactions by using interfaces. The more relevant WS-BPEL 2.0 (Web services BPEL), is an XML based language that allows for specifying business processes in terms of web services [22]. However, WS-BPEL 2.0 does not provide any support for access control, and it is not possible to add user behavior to the business process once it is deployed.

Petri nets have also been used to specify workflows for quite some time [3, 35, 34]. Petrinets use a graphical, state based approach to model workflows. They have good analysis techniques available. However, we have not examined the use of Petrinets for formulating state based access control.

Now, we shall discuss some of the state of the art models for access control and security in workflows and business processes. Role Based Access Control (RBAC) [15] is widely used along with workflows to model access control. Several such models [8, 9, 10] have been proposed, however all of these models are static in nature, and in each of them the access control is not related to the state of the workflow. Workflow Authorization Model (WAM) [5] and it successor SecureFlow [38] are also some models which are used, which are again static in nature. Most of these models are based on petrinets.

SecureBPMN is an implementation for securing BPMN by Brucker et al. [12]. It extends BPMN with a security language called SecureBPM which is based on RBAC. It applies Model driven security to enforce security constraints. Bertino et al. [7] have introduced RBAC-WS-BPEL which is an extension of WS-BPEL added with RBAC and authorization constraints. Wolter et al. [42, 41] have also proposed a model driven approach to model security goals along with the business process.

The popular Content Management System, Drupal also provides a workflow module [36] with functionalities like access control. In this module workflow states are used that have to be manually specified, that can lead to state explosion, thereby imposing a limitation on the number of work- flow states. In our model, we tie states to individual users, and at any point the state of the workflow is given by the combination of the states of all the users. By this, we can easily implement complex workflows with a large number of states.

6. CONCLUSION AND FUTURE WORK

We have given a state based access control approach to model e-governance processes to enable the basic principles of open government. The model uses the process specification to develop e-governance applications, where the citizen can be part of the process leading to transparency, participation and collaboration. This process specification itself is visible to the citizens, thus increasing the transparency of the process. The specification can also be verified and validated at run-time, using various analysis tools available for CCS.

The model introduced in this paper is a simple model that can form the core of several improvements and extensions. The concept of roles needs to be added to the model in order to make it scalable. The model assumes that the workflow does not depend on the field content; however this may not always be the case. This model can also be extended with the concept of private channels defined in π-calculus [26] for security against eavesdropping attacks. It also leaves out the issue of meta-level access control that is concerned with granting privileges to design view specifications. Future extensions should consider incorporating deadlines, exceptions and timeouts, and their specification in e-government workflows.

7. REFERENCES

[1] Defining state based access controls. http://pic.dhe.ibm.com/infocenter/sr/v7r5/ index.jsp?topic=%2Fcom.ibm.sr.doc%2Ftwsr_ configrn_governance24.html. [Online; accessed 2-Aug-2013].

[2] Revisioning with state-based content access control. https://drupal.org/node/408052. [Online; accessed 2-Aug-2013].

[3] N. R. Adam, V. Atluri, and W. kuang Huang. Modeling and analysis of workflows using petri nets. Journal of Intelligent Information Systems, 10:131–158, 1998.

[4] Ankur Goel. Prototype Implementation. https://github.com/ankur1990/impl. [Online; accessed 2-Aug-2013].

[5] V. Atluri and Wei-Kuang Huang. Enforcing mandatory and discretionary security in workflow management systems. Journal of Computer Security, 5(4):303–340, 1997.

[6] D. Beer, S. H¨ohne, H. Petersohn, T. P¨ohnitzsch, G. Ru¨nger, and M. Voigt. Designing a distributed workflow system for e-government. In Proc. of the 24th IASTED International Conference on Modelling, Identification, and Control, CD-ROM, pages 583–588, 2005.

[7] E. Bertino, J. Crampton, and F. Paci. Access control and authorization constraints for WS-BPEL. In Web Services, 2006. ICWS'06. International Conference on, pages 275–284. IEEE, 2006.

[8] E. Bertino, E. Ferrari, and V. Atluri. A flexible model supporting the specification and enforcement of role-based authorization in workflow management systems. In Proceedings of the second ACM workshop on Role-based access control, pages 1–12. ACM, 1997.

[9] E. Bertino, E. Ferrari, and V. Atluri. The specification and enforcement of authorization constraints in workflow management systems. ACM Trans. Inf. Syst. Secur., 2(1):65–104, Feb. 1999.

[10] R. A. Botha and J. H. Eloff. A framework for access control in workflow systems. Information management & computer security, 9(3):126–133, 2001.

[11] R. Breu, M. Hafner, B. Weber, and A. Novak. Model driven security for inter-organizational workflows in e-government. In E-Government: Towards Electronic Democracy, pages 122–133. Springer, 2005.

[12] A. D. Brucker, I. Hang, G. Luckemeyer, and R. Ruparel. SecureBPMN: Modeling and enforcing access control requirements in business processes. In Proceedings of the 17th ACM symposium on Access Control Models and Technologies, pages 123–126. ACM, 2012.

[13] R. Cleaveland, J. Parrow, and B. Steffen. The concurrency workbench: A semantics-based tool for the verification of concurrent systems. ACM Transactions on Programming Languages and Systems (TOPLAS), 15(1):36–72, 1993.

[14] F. Curbera, R. Khalaf, N. Mukhi, S. Tai, and S. Weerawarana. The next step in web services. Communications of the ACM, 46(10):29–34, 2003.

[15] D. Ferraiolo and R. Kuhn. Role-based access control. In 15th NIST-NCSC National Computer Security Conference, pages 554–563, 1992.

[16] G. Ferrari, G. Modoni, and P. Quaglia. Towards a semantic-based verification environment for the pi-calculus. 1995.

[17] A. Goel and V. Choppella. Algebraic modelling of educational workflows. In Technology for Education (T4E), 2012 IEEE Fourth International Conference on, pages 153–156. IEEE, 2012.

[18] Government of India. Public grievance process channels. http://www.pgportal.gov.in/Channels.aspx. [Online; accessed 2-Aug-2013].

[19] Government of India. RTI, Passport Application Process. http://passportindia.gov.in/ AppOnlineProject/pdf/RTI.pdf. [Online; accessed 2-Aug-2013].

[20] Government of India. Right to Information Act. http://rti.gov.in/rti-act.pdf, 2011. [Online;accessed 2-Aug-2013].

[21] J. Habermas. The structural transformation of the public sphere: An inquiry into a category of bourgeois society. MIT press, 1991.

[22] D. Jordan and J. Evdemon. Web services business process execution language, version 2.0. http://docs.oasis-open.org/wsbpel/2.0/OS/ wsbpel-v2.0-OS.pdf, 2007. [Online; accessed 2-Aug-2013].

[23] A. Kamra and E. Bertino. Privilege states based access control for fine-grained intrusion response. In Proceedings of the 13th international conference on Recent advances in intrusion detection, RAID'10, pages 402–421, Berlin, Heidelberg, 2010.

[24] D. Lathrop and L. Ruma. Open government: Collaboration, transparency, and participation in practice. O'Reilly Media, 2010.

[25] R. Milner. Lectures on a calculus for communicating systems. Springer, 1985.

[26] R. Milner. Communicating and mobile systems: the pi calculus. Cambridge university press, 1999.

[27] S. K. Mohapatra, D. K. Das, B. Das, J. P. Bakshi, and S. K. Panda. Workflow based e-governance applications: Case study of WF-Bhulekh. In Towards Next Generation E-government, SIGEGOV, 2006.

[28] National Knowledge Commission. Recommendations on E-governance. http://pib.nic.in/archieve/ others/2006/may2006/nkc20060509.pdf, 2006. [Online; accessed 2-Aug-2013].

[29] United Nations. Millennium development goals.

http://un.org/special-rep/ohrlls/lldc/MDGs.pdf, 2000. [Online; accessed 2-Aug-2013].

[30] A. Ojo, T. Janowski, and E. Estevez. Domain models and enterprise application framework for developing electronic public services. In Proceedings of the 6th International EGOV Conference, Regensburg, Germany, volume 1, pages 157–164, 2007.

[31] M. Pankowska. National frameworks' survey on standardization of e-government documents and processes for interoperability. Journal of theoretical and applied electronic commerce research, 3(3):64–82, 2008.

[32] W. Parks. Open government principle: Applying the right to know under the constitution. The George Washington Law Review, 26:1–22, 1957.

[33] D. K. Punia and K. B. C. Saxena. Managing inter-organisational workflows in e-government services. In Proceedings of the 6th international conference on Electronic commerce, ICEC '04, pages 500–505, New York, NY, USA 2004. ACM.

[34] K. Salimifard and M. Wright. Petri net-based modelling of workflow systems: An overview. European journal of operational research, 134(3):664–676, 2001.

[35] W. M. van der Aalst. The application of petri nets to workflow management. Journal of circuits, systems, and computers, 8(01):21–66, 1998.

[36] J. VanDyk. Workflow module in drupal. http://drupal.org/project/workflow. [Online; accessed 2-Aug-2013].

[37] G. Verginadis and G. Mentzas. A light modelling framework for e-government service workflows. Electronic Government, an International Journal, 1(4):420–438, 2004.

[38] Wei-Kuang Huang and V. Atluri. Secureflow: A secure web-enabled workflow management system. In ACM Workshop on Role-Based Access Control, pages 83–94, 1999.

[39] S. A. White. Introduction to BPMN. IBM Cooperation, pages 2008–029, 2004.

[40] S. A. White. Process modeling notations and workflow patterns. Workflow Handbook, 2004:265–294, 2004.

[41] C. Wolter, M. Menzel, and C. Meinel. Modelling security goals in business processes. Proc. GI Modellierung, 127:197–212, 2008.

[42] C. Wolter and A. Schaad. Modeling of task-based authorization constraints in BPMN. In Business Process Management, pages 64–79. Springer, 2007.

[43] W. Yi, P. Pettersson, and M. Daniels. Automatic verification of real-time communicating systems by constraint-solving. In FORTE, volume 6, pages 243–258, 1994.