Specification and Modelling of Workflow Management Systems with State Based Access Control

Thesis submitted in partial fulfillment
of the requirements for the degree of

M.S. (by Research)
in
Computer Science

by

Ankur Goel
200702008
ankur.goel@research.iiit.ac.in

International Institute of Information Technology, Hyderabad
(Deemed to be University)
Hyderabad - 500 032, INDIA
April, 2016

International Institute of Information Technology
Hyderabad, India

CERTIFICATE

It is certified that the work contained in this thesis, titled "Specification and Modelling of Workflow Management Systems with State Based Access Control" by Ankur Goel, has been carried out under my supervision and is not submitted elsewhere for a degree.

_____ _____

Date Adviser: Prof. Venkatesh Choppella

To my family

Acknowledgments

Firstly, I would like to acknowledge my professor, Prof. Venkatesh Choppella for his guidance and support throughtout my years with him. I would like to thank everyone from SERC, with whom I have had so many brainstorming sessions. Most importantly, I would like to thank my family, specially my mom and dad for their complete trust and continous support throughtout my life. This would not have been possible without them. Lastly, I would like to acknowledge my “dulla” batchmates, who have taught me so much in so many aspects of life.

Abstract

A Workflow is a collection of coordinated tasks designed to carry out a well-defined process. Workflows are ubiquitous in process management. The question we address in this thesis is how to specify and design verifiable workflows for such processes. To specify workflows, we borrow a simple algebraic notation from computer science. We illustrate the use of two algebraic specification languages: Pi-calculus and Calculus of Communicating Systems (CCS), through a series of typical workflow examples.

We explore two areas of application for workflows: Education and E-governance. Building such 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 systems carry clear specifications of how access to users' documents are managed throughout an 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. The data store consists of fields and forms. The workflow is specified as concurrent processes each representing a user. The access control layer specifies, for each user (process), the user's 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 give a web-based prototype implementation for the model and show an example of an application generated by the implementation. This implementation differentiates access control from the workflow engine. However, we also show how access control can be modelled using Pi-calculus.

Contents

List of Figures

List of Tables

Chapter 1

Introduction

Workflow Management Systems (WFMS) have been gaining popularity over the years. This has been driven by the need for new high level approaches for developing web based applications. By workflows, we mean a collection of coordinated tasks designed to carry out a well-defined process typically involving multiple identities like teachers, students, administrators, employees etc. A workflow separates the various activities of a given organisational process into a set of well-defined tasks [32]. The main idea behind these systems is to provide a high level environment to model and automate various processes. In a workflow, various tasks are carried out by several users in accordance with a set of rules which is the model of the workflow.

A workflow management system (WFMS) is a generic software tool which allows for the definition, execution, registration and control of processes [51]. A process is a sequence of one or more related and structured activities that have a clearly stated objective. It can be something as simple as conducting an exam for a student, to something as complex as the Passport Application system for the government of India. Because processes are such a dominant factor in workflow management, it is important to use an established framework for modelling and analysing workflow processes [51].

WFMS address a real world problem of expressing complex processes as entities. This has lead to a number of commercially available WFMS, such as Taverna [10], an open source domain independent WFMS for scientific workflows, Comindware [1], a software for Adaptive Business Process Management and Workflow Automation, and ProcessMaker [7]. IBM Lotus [3] also provides support for automating workflows. Most of the focus of the market has been to provide tools for expressing these processes using graphical tools. This is primarily due to the ease of use of these tools for the end user. However, a formal modelling based approach for expressing these processes has its own merits.

Formal modelling of workflow processes is not an unexplored area, several techniques have been used to model workflows in WFMS. A graphical notation, Petri nets [29] are widely allocated for modelling workflows, as they feature a state-based approach towards modelling. The Petri net was invented by Carl Adam Petri in the sixties [29]. Petri nets and other modelling based approaches provide formal semantics, expressiveness and various analysis techniques, that are unavailable in most commercial WFMS.

However, the idea of specifications using a graphical model leaves scope for improvement. The idea of an algebraic approach that can be published as a specification in a widely available documentation. In their paper, Smith and Finger [68] introduced Pi-calculus as one such approach that can been used for modelling workflows. Calculus of Communicating Processes or CCS [54] is another algebraic approach that is used to model workflows, however it should be noted that Pi-calculus is essentially an extension of CCS.

Another major key flaw that we observe in modern commercial WFMS is the absence of any form of access control. Workflows are often used to model highly sensitive processes, such as E-governance processes, where the user data is confidential. We believe that it is important to have an access control policy such that even though no user can change the access control policy, it is open to all the users for scrutiny. This should also be published as a part of the specification of the WFMS. Such a specification can later be used for a security audit, and even the verification of the system. We discuss more the about the different access control models and their extensions into WFMS in the next section.

This thesis proposes an approach for algebraic modelling of workflow processes, also taking into consideration the access control of the users involved. We use the concept of State Based Access Control, where the system state evolves by user interaction. Also, for every user, access to shared data in the workflow changes with every interaction. We employ an algebraic formalism such as CCS or Pi-calculus and to specify the workflows layer. These formal languages, allow for static and dynamic analysis of such workflows [24, 31, 84]. We use a layered approach with layers as: Process Layer, Data Layer and View Layer. The data layer in the model is stored in form of fields and forms. The process layer expresses the workflow. 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. We show later how this approach can then also not only for verification and analysis of workflows, but also to build a web application just based on the specification.

Commercially available WFMS are used in almost all domains, ranging from scientific processes to E-governance. As a part of this thesis, we focus on two areas that, according to us, can be best used to demonstrate our approach. Firstly, we consider E-Governance, where most of the processes are well described and their plain-text specifications are mostly available. Workflows have been modelled for several E-gov processes [16, 56] previously. The other area is Education, where processes like conducting an exam, grading assignments etc are few of the most intuitive examples to represent workflows. The next sections discusses these two areas.

1.1 Domains for application of Workflows

1.1.1 Workflows in Electronic Governance

The right of the citizen to scrutinize and participate in the process of governing forms the basis for Open Government [62]. The origins of the concept of Open Government date back to the era of

European Enlightenment [37], whereas the term was formally introduced in the United States of America [62]. 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 [50]. 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 [50]. 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) [36] 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) [58] also recognize the need for open government. Goal 8 expresses the need for developing "open, rule-based, predictable and non-discriminatory" financial and trading systems. However, this would be even better if it could be generalised to all systems. In recent years, many governments in the world have moved towards standardisation of e-governance processes to align themselves with the principles of open government [57, 60]. However, in order to make this vision a reality, governments must ensure that specifications of their process workflows are published and available to all for scrutiny. These specifications can then form the basis of verifiable and auditable implementations. The problem here is close to that found 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 thesis, 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 [16, 59, 63, 73], 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 [2] and Drupal [9]. 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 then the state of a user. Another approach is Privilege State based Access Control(PSAC) [42], 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.

1.1.2 Workflows in Education

Most educational institutions today are growing at a rate like never before with many evolving into large scale virtual organizations. This impacts the requirements of educational software in terms of the scale and complexity. One component of this complexity is the type of complex workflows that capture educational processes.

Any implementation of enterprise resource planning (ERP) for educational systems will need to model such complex workflows. Conducting an exam is a workflow, even taking attendance can be expressed as a workflow. In simple words, it is the "flow of work", which we try to formally specify for such workflows.

In this thesis, we discuss the applications of workflows in the field of education with a simple example of the exam workflow. The student and the teacher are the two roles that make this workflow. The teacher sets the paper, gives it to the student, who submits the solution to the teacher, who then gives the grade back to the student. For simplicity, we assume without loss of generality that the exam consists of just one question and one corresponding answer.

Access control is again essential in this example, as no teacher would like any student to have access to the exam paper. Also similarly a student would not like any other student to have access to his or her grade.

1.2 Flow of the Thesis

In this chapter we gave a brief introduction to workflows and their applications in E-governance and education. Chapter 2 gives an idea on the related work in the fields of workflows and access control. This chapter first discusses the different access control models in collaborative systems, and then the various

models for several WFMS. In Chapter 3, we propose an approach to algebraically model workflows and also give a brief introduction to CCS and Pi-calculus.

Then in Chapter 4, we show via examples from the field of E-governance and education that how this model can be applied. In chapter 5, we discuss a web based implementation for our approach, that generates an access controlled web application, based on the specification of the model. We also compare our implementation with other similar ones. In Chapter 6, we give a theoretical approach that could be used for modelling access control in workflows using Pi-calculus. Finally, chapter 7 concludes the dissertation with pointers to future work.

Chapter 2

Related work on Access Control in Workflows

Workflows are often used in sensitive domains such as finance and E-governance. This raises the issue of highly sensitive data being involved as a part of the workflow process. Thus, security is a key concern when designing workflow management systems. Controlling the access is essential to ensure the security of the system. Access control, as the name suggests, is the ability to control the access to the data for all the users of the system.

In our view of access control, we consider a system having many subjects (such as users, groups, organizations) and objects (data). At any point of time a subject should logically be able to be mapped to every object with an access level. Access levels are the different levels of accesses that a user may have for an object.

Separation of duty (SoD) is also a key requirement for building secure systems. SoD aims at reducing the risk of fraud by not allowing any subject to have sufficient and absolute authority within the system to perpetrate a fraud on his own. E.g. Opening a safe requires 2 keys held by different people with different roles.

Business processes are closely related to workflows. Thus, we also look at access control from the perspective of business processes. A business process is a set of one or more linked tasks or activities that collectively realize a business objective or policy goal. BPMN (Business Process Modelling Notation) [78] is one such graphical modelling notation for modelling business processes. WS-BPEL (Web Services Business Process Execution Language) [27, 40] is also an XML-based workflow process language, which provides a syntax for specifying business processes based on web services. For a comprehensive comparison of business process methods, the reader may refer to Wang et al[75].

The road map of the chapter is as follows. In section 1, we discuss about general access control approaches in collaborative environments. In section 2, we see how these approaches are extended to workflows by looking at several workflow models, with various access control approaches employed in each of them. Section 3 talks about access control in business processes. Finally in section 4, we give the conclusion of our survey with our understanding of the shortcomings in the current approaches.

2.1 Access Control in collaborative environments

In this section, we discuss various access control approaches applicable to collaborative environments. The reader should skip this section if they are only interested in access control related to WFMS. We compare various access control approaches on several factors. The key factors are the security and flexibility they offer. We also analyze them on the separation of duty (SoD), contextual and temporal constraints.

2.1.1 MAC (Mandatory access control)

It is defined by the Trusted Computer System Evaluation Criteria [64] as "a means of restricting access to objects based on the sensitivity (as represented by a label) of the information contained in the objects and the formal authorization (i.e., clearance) of subjects to access information of such sensitivity". It is a class of access control models where there is a system-wide policy which controls access of various users and no user can change that access control policy. This model is not very flexible and the access control policy has to be defined by the system, however this fact ensures that the security provided is absolute and the security of the system is easily verifiable.

2.1.2 DAC (Discretionary access control)

It is defined by the Trusted Computer System Evaluation Criteria [64] "as a means of restricting access to objects based on the identity of subjects and/or groups to which they belong. The controls are discretionary in the sense that a subject with a certain access permission is capable of passing that permission (perhaps indirectly) on to any other subject". In this model, a user is responsible for controlling the access of the object he/she has ownership to, and can grant other users access to such objects. This model is very flexible because the user can control the access of various objects. But this flexibility comes at the cost of security as a malicious or flawed user can even change permissions of some objects that may effect other users of the system.

DAC together with MAC are the two basic approaches for access control. The following models use either one or both of these approaches to model access control.

2.1.3 LBAC (Lattice Based Access Control)

LBAC is an extension of MAC wherein a lattice is used to define the access of each and every subject. It was introduced by Denning [28] and later used by Sandhu [66]. It makes use of the Bell-LaPadula (BLP) model for access control similar to DAC. In mathematical terms, the lattice is represented as a partial order set. If two subjects want to access an object, the access is the meet of the levels of them, and if two objects are combined to form another object, its level will be the result of join of both their levels. It shares all the weaknesses and strength of MAC.

2.1.4 Access Matrix Model

The Access Matrix model maps the access rights of each subject to every object in the system. It was first introduced by Butler W. Lampson in 1971 [49]. It is sometimes implemented as a matrix of "subjects X objects", and each entry is the access level of the subject for the corresponding object. However generally it is implemented in the form of ACL's (Access Control Lists). Each object has a corresponding ACL, which stores, for each subject in the system, the access rights the subject has on the object. Capability lists are also an approach used where each subject is associated with a list, known as the capability list. This list stores each object in the system along with the access rights the subject has on the object.

This model is not able to define the rules by which permissions change in the system, and therefore only gives an incomplete description of the system's access control security policy. Also there is a huge overhead on space in this model. This model is good for mapping static permissions, however ACL- and capability-based approaches lack the ability to support dynamic changes of access rights. Also more complex access policies such as least-privilege or conflict-of-interest are difficult to be mapped in this model. Access rights related to the context of the system also cannot be modelled by this model.

2.1.5 RBAC (Role Based Access Control)

RBAC was introduced by Ferraiolo and Kuhn [30]. Various models were given [67] for RBAC which could implement both MAC and DAC. The basic idea behind RBAC is that permissions are mapped to roles instead of subjects like in the previously discussed models. Roles are assigned to subjects based on their functionalities and a subject can have multiple roles. Each role is assigned specific permissions, based on which access is controlled. For defining RBAC we need to assign subjects to roles and then assign access rights for the objects to the roles. By assigning subjects to roles, this model provides a powerful means of enforcing conflict of interest and cardinality rules for roles [61].

RBAC differs from the Access Matrix model as in that it assigns permissions to specific operations with meaning in the organization, rather than to low level data objects. RBAC has been shown to be particularly well suited to separation of duties requirements. The major problem with RBAC is that, like the previous access control models, it is still static in nature. Roles are generally predefined, and cannot change easily with contextual changes in the environment. Also because of assigning subjects to roles, we lose control on individual subjects which may be required in several cases. Besides, specific constraints are not easily mappable with RBAC.

2.1.6 TBAC (Task Based Access Control)

TBAC [69] models access controls from a task-oriented perspective rather than the traditional subject-object perspective as used in the previously defined models. This model is used for defining "active" security models. Instead of having a system-centric view of security, TBAC approaches security modeling and enforcement at the application and enterprise level. The TBAC model includes domains that

contain task-based contextual information. Access control granted is directly related to the progress of tasks. Each step in the task is associated with a protection state containing a set of permissions. The contents of this set change as per the task that is happening.

Because permissions for a user in TBAC change as per the progress of tasks, contextual information can be considered in access control unlike RBAC. This also resembles the notation followed in WFMS, which will be discussed in the next section. However such a model can only be used if the entire system can be divided into simple tasks. Several complex systems cannot be modelled using this. Also Just-in-time permission assignment can lead to various constraints such as race conditions across workflows.

2.1.7 Team Based Access Control (TMAC)

TMAC is an approach applying RBAC in collaborative environments [70]. A team is an abstraction that encapsulates a collection of users in specific roles with the objective of accomplishing a specific task or goal. It defines two important aspects of the collaboration context, user context and object context. User context provides a way of identifying specific users playing a role on a team at any given moment, and object context identifies specific objects required for collaboration purposes. This allows for fixing the weakness of RBAC and specify control for individual subjects/objects. C-TMAC [33] has also been introduced which incorporates contextual aspects into TMAC.

2.1.8 Organisation Based Access Control (OrBAC)

In order to specify a security policy, the OrBAC model [26, 41] introduces the concept of organizations. An organization is any active entity that is responsible for managing a security policy. Each organization can define its proper policy using OrBAC. Then, instead of modelling the policy by using the concrete implementation-related concepts of subject, action and object, the OrBAC model suggests reasoning with the roles that subjects, actions or objects are assigned to in an organization. The role of a subject is simply called a role as in the RBAC model. The role of an action is called activity and the role of an object is called view. Each organization can then define security rules which specify that some roles are permitted or prohibited to carry out some activities on some views. Particularly, an organization can be structured in many sub-organizations, each one having its own policy. It is also possible to define a generic security policy in the root organization. Its sub organizations will inherit from its security policy. Also, they can add or delete some rules and so, define their proper policy.

RBAC can only model static security requirements whereas OrBAC provides means to define dynamic and contextual security requirements. It uses the concept of organizations, where if workflows are defined within different organizations, flows between different organizations need to be managed. OrBAC uses contexts to express permissions that can apply in special cases. Provisional context is one where permissions depend on previous actions performed on the system. The temporal context, on the other hand defines temporal constraints to help define dynamic temporal rules.

2.1.9 Summary

It can be seen that most of these models have a compromise between security and flexibility. Generally one is provided at the cost of the other. Hence, there is a need for a model which can offer one, without compromising the other in a major way. Also it can be seen that to enhance security, some models handle SoD, Contextual and Temporal concepts. Here is a comparison of the model discussed earlier, with how the perform with these factors.

2.2 Access Control in workflow environments:

The preceding section discussed the access control models in collaborative environments. Most of these models, have been incorporated along with WFMS to provide access control. Now we discuss some of these models that are being widely used:

2.2.1 Workflow Authorization Model (WAM)

WAM [15] dynamically assigns authorizations to support workflow activities in a way that the time interval associated with the required authorization to perform a task changes according to the time during which the task actually executes. WAM uses the notion of an Authorization Template (AT) which specifies the static parameters of the authorization that can be defined during the design of the workflow. ATs are attached to tasks. When the task starts execution, its AT(s) is used to derive the actual authorization. When the task finishes, the authorization is revoked. A system called SecureFlow [76] is built on this. However this model does not consider the order of execution of tasks for the same object as well as the authorization access to objects. This model is also static is nature, and adequate only for centralized management of workflow security.

2.2.2 RBAC based WFMS

Bertino et al. first proposed this model [18, 19] to extend RBAC into workflow management systems. It is an access control mechanism that enforces the security policy of the organization, typically expressed as a set of authorizations. The authorizations here can be expressed in terms of roles. As described in the previous section, roles represent organizational agents to perform certain job functions within the organization.

There are several advantages which come with using RBAC in WFMS. Most of these are inherited from the RBAC model that is used in collaborative environments. Firstly, it reduces the extra space overhead requirements due to access control because the number of roles are much smaller than the number of individuals. Secondly, the use of roles as authorization subjects, instead of users, avoids the need of having to revoke and regnant authorizations whenever users change their positions and/or duties within the organization. Role-based authorization is particularly beneficial in workflow environments in

facilitating dynamic load balancing when a task can be performed by several individuals. Also, RBAC is widely used in the development of WMFS's access control. The predominance of RBAC is due to its being able to express business functions in terms of roles and the user can switch flexibly among different roles. In addition, the roles makes the users being awarded the least privileges. Therefore, RBAC has been the most natural choice in the authorization mechanism of WFMS.

However, this model still had its weaknesses. RBAC is unable to model authorization constraints on roles. Seperation of duties is also not possible using RBAC. However, because a workflow decomposes a complex activity into a number of smaller well-defined tasks, separation of duties could naturally fit into workflow models. Another major problem is that the workflow and permissions are defined separately. This problem arises in cases where the permissions depend on the state of the workflow itself. Take an example of an educational WFMS for a school, which has two roles, student and teacher. The workflow is such that a student needs to take an exam, and after he/she has submitted the exam, he/she should not be able to change it. This implies that for different students the permission of the "view exam" task is different. This cannot be modelled in existing RBAC based WFMS.

There have been several models based on RBAC. Ahn et al. [13] has used this approach and injected it into a web based workflow system. Wainer et al. [74] has used RBAC to address the need for controlled overriding of constraints. However these models face the same issues as those faced in the RBAC based system.

2.2.3 CoSAWoE framework

CoSAWoE stands for Context-sensitive access control in workflow environments. Botha and Eloff [21] provided this framework for access control in workflow systems based on RBAC. They discussed access control and workflow management as two different spheres and have proposed a RBAC based access control model to combine both spheres into a single solution. This model takes the functional requirements (such as order of events, strict least privilege, Seperation of duty(SoD)), the organization structure, and the workflow into one model. CoSAWoE acts as a layer connecting both the access control sphere and the workflow run time environment, with session control at run-time. In a session, a subject is linked with the roles that are granted to it. Thus it receives only a subset of the permissions associated with the roles that it may assume.

This framework tries to connect access control with workflows, but it is only done through the user sessions. The specification of the workflow and the specification of access control are defined as separate tasks and combined via a framework. Thus this model still suffers from all weaknesses of RBAC based workflow models. The authors view access control specification and workflow specification as separate entities.

Botha extended this framework into his thesis [20] in 2008. The mechanism was implemented for XML and agent based environments as a part of the thesis.

2.2.4 Petri net based models

Petri nets are widely allocated for modelling workflows [12, 72, 65]. Petrinets use a graphical, state based approach to model workflows. They have good analysis techniques available. Here we discuss some models using petri nets to model workflows.

Atluri and Huang [77] proposed a model which is an extension of WAM to support roles and authorization constraints such as Seperation of duties etc. To ensure that these tasks are executed by authorized subjects, and to make sure that authorized subjects gain access on the required objects only during the execution of the specific task, granting and revoking of privileges need to be synchronized with the progression of the workflow through proper authorization mechanisms. This model uses Colour Timed Petri Nets (CTPN) and Open Petri Nets (OPN) to express workflows, and addresses the temporal aspects of workflows as well. The authors have used simulations, reachability trees and matrix equations to do their safety analysis. In this model, permissions change as tasks are executed, but fine grained access to individual tasks is not possible. However, it may be possible that permissions change while a task is in progress.

Knorr proposed a model for dynamic access control through petri net workflows [44]. This model uses an access control matrix to grant subject privileges to objects. Access rights are granted according to the state of the workflow. This paper has shown how access rights can be derived from a petri net dynamically. A subject has only the access rights which are needed for 'activated' activities. This model dynamically assigns access based on the activity that is being performed. This model was extended with multilevel security later [45]. He has also analyzed the model for SoD constraints [47]. This was extended for the World Wide Web in [46].

Luo et al. also propose a petri net based workflow access control module named WACM (workflow access control model) [52]. This model also supports dynamic role based authorization. They define the architecture for building such models. An Access control matrix is used to implement this model.

2.2.5 Access Control Mechanisms for Inter-organizational Workflow

SALSA [43] proposes a new kind of security architecture. This model separates the inter-organizational workflow security from concrete organization level security enforcement, and the enforcement of fine-grained access control for inter-organizational workflow. It uses a combination of two different kinds of access control modules in the overall security architecture [43]:

- An organization-specific access control module (OACM)that is controlled by each organization and enforces a security policy that was set by each organization.
- The task-specific access control module (TACM) that is controlled by each workflow and enforces task-specific security policies. Only a person with intimate knowledge of the workflow can set the security policy of each task because, in general, a task-specific security policy depends on the semantics of the workflow. The purpose of the TACM is to provide fine-grained access control for both subject and object.

2.2.6 Access Control in WFMS based on Tasks and Roles (TRBAC)

This model [39] uses both tasks and roles in order to meet the requirements for modern enterprise environments. In a WFMS, tasks are the fundamental units of workflow. In TRBAC, there are responsible tasks for each role and the permissions of objects are bound with specific tasks. TRBAC supports the "active" access control through binding the permissions on tasks and sustains the "passive" access control by grouping users into roles. TRBAC binds permissions on tasks and groups users operating the same tasks into roles, thus giving both the advantages of TBAC and RBAC. In this model tasks are assigned to roles, and then permissions of the objects are assigned to tasks. The problem with this model is that any task will have the same permissions throughout its execution, however there can be a need for permissions to change within a task, which cannot be specified using this model.

2.2.7 Other models

Botha et al. [22] propose a model to solve the problem of addressing SoD constraints in workflow systems.

Wu et al. propose METEOR [82] which is also an access control model based on RBAC but it works on security of application data. They model a healthcare workflow application, and use UML to model workflows.

Zhao et al. propose IRBAC [85] which enables users to express business character and phase authorization constraints. There are several other models like Temporal-RBAC which introduce temporal aspects into workflows, however being out of scope of our work, have not been studied.

2.3 Access Control for business processes

A business process is a set of one or more linked tasks or activities that collectively realize a business objective or policy goal. Access control in business processes is still a relatively new topic and there is limited literature available in this area. In this section we discuss the access control approaches primarily used in BPMN and WS-BPEL.

2.3.1 Access control in BPMN

Business process Model and Notation (BPMN) is a graphical notation for specifying business processes in terms of a Business process diagram (BPD). It is being used vastly to model business processes. The reader may refer to [78, 79] for more information about BPMN. As in workflows, there is a need to model access control in BPMN. Below are some models that provide access control to BPMN.

Wolter et al. [81, 80] have proposed a model driven approach to model security goals along with the business process. The goals focused in this paper are : Confidentiality, Integrity, Authentication, Authorisation, Traceability & Auditing and Availability. They use Attribute Based Access Control (ABAC) for

implementing access control. According to ABAC, the access control decision is made based on subject attributes, resource attributes, and attributes of the resources environment. The attributes that must be present are specified by the policy constraint called "Permission". This model breaks the development into various layers: Integration layer, Organisational layer and the Business Process layer in the middle. The Integration layer contains the data, resources and the services provided. The organisational layer holds the structure of the organisation with the roles of the corresponding users. The business process layer maps the business process with the dependencies of the other two layers.

SecureBPMN is another implementation for securing BPMN by Brucker et al. [23]. This paper discusses how Access Control, Separation of Duty(SoD), Binding of Duty and Need to Know constraints are modelled and enforced in business process driven systems. They extend BPMN with a security language called SecureBPM which is based on RBAC. They apply the MDS (Model driven security) paradigm to enforce security constraints. The requirements are automatically translated into XACML.They use Policy Decision Point (PDP) to decide if a request should be granted or not. If a request is denied, the PEP in the user interface of the workflow management systems informs the user about the violation of the security policy.

2.3.2 Access control in WS-BPEL

Bertino et al. have talked about access control and authorization constraints for WS-BPEL [17]. This paper introduces RBAC-WS-BPEL which is an extension of WS-BPEL added with RBAC and authorisation constraints. They also address separation of duty(SoD) and binding of duty(BoD) constraints. For this they have also introduced BPCL (Business process constraint language) to articulate authorisation constraints. They use XML based XACML for the specification of RBAC.

An RBAC-WS-BPEL permission is a tuple (A , Action) where A is the identifier of an activity in BP and Action identifies the type of action that can be performed on activity A. A RBAC-WS-BPEL authorisation schema for BP is a tuple (R, P, RA) where R is a partially ordered set of roles associated with BP, P is the set of permissions defined for the activities in BP and RA, a subset of roles mapped with their permissions, is a role-permission assignment relation. They provide role authorisation constraints as well as user level authorisation constraints so as to be able to deal with fine grained access control. They use XACML to model this system. BPCL specifies authorisation constraints that apply to activities in a business process. This can be used to specify priority of activities so as to model SoD and BoD constraints.

2.4 Conclusion

In this survey we have studied and analyzed various access control approaches in WFMS. We have pointed out their strengths and their weaknesses. A general finding of the survey is that Access control and workflow control are thought of as two independent problems, and a layer is modelled by most

authors to interconnect the two problems. There are also some approaches based on petri nets that offer state based dynamic access control, however they have their own flaws. One key observed flaw is that they provide a graphical representation of the workflow, however an algebraic specification would be better suited for verification and publication of the workflow.

Also it can be seen that there is a need of model which is so flexible that access can be defined at every step and still secure that its access control is guaranteed. Another requirement is to reduce the space overhead required for access control. Roles seem like a great approach towards this, however they come at the cost of fine-tuned access control. It appear to us that states seem to be the best way to define the different accesses an user may need. In the next chapter, We map the state of a user to the access over different objects as a part of the proposed model.

Chapter 3

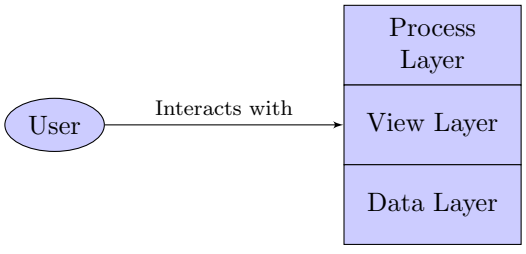
A layered model for specifying state based access control in workflows

3.1 An Approach towards Modelling Access Control for Workflows

This chapter introduces a 3-layered approach to model access control in workflow-based systems, discussed in the last chapter. The roots of our approach lie deep in one of most important fundamentals of software engineering, i.e. the idea of formal specifications for system development. These specifications can be used to describe a system, to analyze its behaviour, and to aid in its design by verifying key properties of interest through rigorous and effective reasoning tools [38].

The proposed approach aims at building a system where the access control policies are secure yet flexible. We employ State Based Access Control that was discussed in the last chapter. State Based Access Control limits the access to the data for a user based on the user's state. The access control specification can also be exported along with the workflow specification, and used for verification purposes. We make use of a very simple relational data store to store the user data, that can be realised by any modern database system.

We consider each user of the system as an individual process, interacting with the other user processes, which work together in order to drive the workflow. We use a formal specification language in order to define the workflow, and add access control using the concept of views derived from the MVC framework [48] to control access to a common data store. The view of a user is a function of its process state.



We model our approach using a layered model inspired from the MVC architecture [48]. The model consists of three layers: a data layer for the data, a process layer to express workflows, and the view

layer for access control. 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.

The rest of the chapter describes the model and its different layers, and we conclude with a brief summary of our approach.

3.1.1 Data Layer

The model assumes a forms-based document store, thus the data layer has forms, fields and users. A Form is a document, which is a collection of fields. Fields are the basic data units, that consist of a label and the field content. However it should be noted that fields are not statically assigned to forms by the data layer. In our model, 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

$$\begin{aligned} \mathit{User} &: \mathit{TYPE} \\ \mathit{Form} &: \mathit{TYPE} \\ \mathit{Field} &: \mathit{TYPE} \end{aligned}$$

The data model operates on a fixed set of users, forms and fields. The users are the subjects for the workflow, and the fields form the objects.

$$\begin{aligned} \mathit{users} &: \mathit{set}[\mathit{User}] \\ \mathit{forms} &: \mathit{set}[\mathit{Form}] \\ \mathit{fields} &: \mathit{set}[\mathit{Field}] \end{aligned}$$

It should be noted that we use a simple model to represent the user data, hence the data layer can be easily realised using any modern database, or even file storage.

3.1.2 Process Layer

The process layer describes the workflow of the system. Each 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 process algebras, specifically CCS and π-calculus, which are briefly introduced below.

3.1.2.1 Introduction to CCS

We use the Calculus of Communicating Processes (CCS) to specify workflows for our model. For a more leisurely introduction, please refer to Milner [54]. Figure 3.1 captures the syntax of CCS. Readers already familiar with CCS may skip this introduction. An action is either silent τ, or a send $\overline{a}$, or a receive a on a channel a. The concurrent composition of processes P and Q is denoted $P|Q$. $\Sigma_i \pi_i.P_i$

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

Figure 3.1: Syntax of CCS.

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 3.2. The first rule, called reaction, allows a process expression with two concurrent processes $\overline{a}.P + M$ and $a.Q + N$, one with a send capability and the other with a receive capability to evolve into another pair of concurrent processes $P|Q$. 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 3.2: Dynamics of CCS rules defined via reduction.

3.1.2.2 Introduction to Pi-calculus

π-calculus is an extension of CCS to incorporate dynamic and private channel passing. The figure below captures the syntax of π-calculus. The reader may also refer to Milner's book [55] for a detailed introduction to π-calculus.

π-calculus has the ability to pass values over channels. $\vec{x}$ denotes a vector $x_1, \ldots, x_n$ of elements. Similar to CCS, an action is either silent τ, or a send $\overline{a}\langle\vec{x}\rangle$, or a receive $a(\vec{x})$ on a channel a. The parameters $\vec{x}$ in process definitions and receive actions can refer to arbitrary (but non-process) values. Angular brackets or parentheses around empty sequences of parameters or values are dropped. Thus,

Actions	π	:=		
			$a(\vec{x})$	Receive
		|	$\overline{a}\langle\vec{x}\rangle$	Send
		|	τ	Silent action
Processes	P	:=		
			0	Empty process
		|	$\Sigma_i \pi_i.P_i$	Guarded Summation
		|	$\mathbf{new}\ a\ P$	Restriction
		|	$P \mid Q$	Parallel composition
Definitions	D	:=		
			$A(\vec{x}) \stackrel{\text{def}}{=} P$	x_i are distinct

Figure 3.3: Syntax of π-calculus.

the send action $\overline{a}\langle\rangle$ is written as $\overline{a}$. $\mathbf{new}\ a\ P$ introduces the channel a local to the process expression P. 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. Rest of the behaviour of π-calculus is similar to CCS.

The dynamics of processes is defined by the basic reduction rules shown in Figure 3.4. The basic two rules are the same, however as value passing is allowed on channels, a reaction takes place differently. It allows a process expression with two concurrent processes $\overline{a}\langle\vec{v}\rangle.P + M$ and $a(\vec{x}).Q + N$, one with a send capability and the other with a receive capability to evolve into another pair of concurrent processes $P|Q[\vec{v}/\vec{x}]$ with the values $\vec{v}$ in second process replacing all occurrences of $\vec{x}$. Note that the alternatives M and N are discarded, here also.

$(\overline{a}\langle\vec{v}\rangle.P + M) \mid (a(x).Q + N)$	$\rightarrow$	$P \mid Q[\vec{v}/\vec{x}]$
$\tau.P + N$	$\rightarrow$	P

Figure 3.4: Dynamics of π-calculus rules defined via reduction.

3.1.2.3 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 chapter shows how this is done with the help of a few 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.

3.1.3 View Layer

The view layer acts as an interface between the data layer, the process layer and the users. The view layer makes use of "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 approach.

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.

$$\textit{Perm}: \qquad \textit{TYPE} \quad = \quad \{--, r-, -w, rw\}$$

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.

$$\begin{array}{lcl} \textit{field-view} & : & \textit{fields} \rightharpoonup \textit{Perm} \\ \textit{form-view} & : & \textit{forms} \rightharpoonup \textit{field-view} \\ \textit{user-view}(\mathrm{u}) & : & \textit{States}(\mathrm{u}) \rightarrow \textit{form-view} \end{array}$$

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.

Chapter 4

Modelling some Example Workflows

4.1 Example from Education: Exam Workflow

This section demonstrates our approach with a simple and intuitive example from the field of education, the exam workflow. We start with a simple model of a student and a teacher, and extend it a case where there are a number of students. We employ π-calculus in order to specify the process layer for this example. This is done in order to enable value passing over channels.

$$
\begin{array}{lcl}
\textit{s-ready} & \stackrel{\text{def}}{=} & \textit{paper}.\ \textit{s-writing} \\
\textit{s-writing} & \stackrel{\text{def}}{=} & \overline{\textit{submit}}.\ \textit{s-waiting} \\
\textit{s-waiting} & \stackrel{\text{def}}{=} & \textit{grade}.\ \textit{s-done} \\
 & & \\
\textit{t-ready} & \stackrel{\text{def}}{=} & \overline{\textit{paper}}.\ \textit{t-waiting} \\
\textit{t-waiting} & \stackrel{\text{def}}{=} & \textit{submit}.\ \textit{t-grading} \\
\textit{t-grading} & \stackrel{\text{def}}{=} & \overline{\textit{grade}}.\ \textit{t-done}
\end{array}
$$

Initial Process Expression:
$\textit{s-ready} \mid \textit{t-ready}$

Figure 4.1: Exam Workflow for one student and one teacher - Process Layer

4.1.1 Simple use case: A Student & A Teacher

This case considers the scenario where there are just two users in the system, a student and a teacher, who interact with each other in order to advance the workflow.

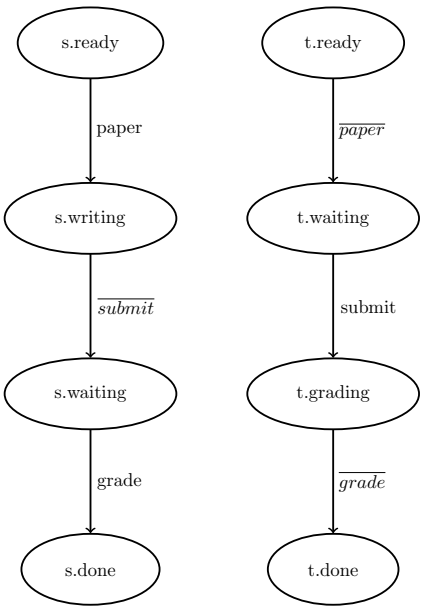


Figure 4.2: Graph for a Student & a Teacher participating in a one-student, one-teacher exam workflow

4.1.1.1 Process Layer

Figure 4.1 shows the process workflow for this example. At any given time, the student is in one of the following possible states: ready to take the exam *s-ready*, writing the exam *s-writing*, waiting for the grade *s-waiting*, and done, *s-done*. The teacher is in one of the following states: ready with the exam, *t-ready*, distributing the exam paper and waiting for the student to submit the exam, *t-waiting*, grading the exam, *t-grading*, and done, *t-done*.

Figure 4.2 demonstrates the workflow for a student (left) and a teacher (right). The nodes of the graph are the states, and the arrows are the transition between states. The labels on the arrows are events, τ denotes a silent transition caused autonomously. Each process proceeds to the next state on a send/receive action only when there is an interaction with another process executing a complementary receive/send action respectively.

Let us consider the case of a student, who starts at state, *s-ready*, receives the paper on the channel, and transitions to state *s-writing*. On sending the submission on the channel, he/she moves to the next state, *s-waiting*, and on getting the grade (signal) on the channel, finishes at the state *s-done*. The teacher has a very similar graph, refer to Figure 4.2 for it. The specification of this interaction is expressed by the defining process equations and an initial process expression.

It is noticeable that this simple workflow is deterministic. There can be only one way in which it can proceed.

$$\textit{s-ready} \mid \textit{t-ready}$$
$$=$$

paper. *s-writing* | $\overline{paper}$. *t-waiting*

$\Rightarrow$

s-writing | *t-waiting*

$\Rightarrow$

...

4.1.1.2 Data Layer

The data model can be seen in Figure 4.3. There are three fields: *question*, *answer*, *grade*. There is just one form: the *exam* form.

users	$=$	$\{s, t\}$	
forms	$=$	$\{ex\}$	Exam Form
fields	$=$	$\{question,$	Question
		$answer,$	Answer
		$grade\}$	Grade

Figure 4.3: Exam Workflow for one student and one teacher - Data Layer

4.1.1.3 View layer

Now we define the view layer for this example. The view associates user states with form view. Figures 4.4 and 4.5 capture the state-based access control for the student and teacher respectively.

s-ready	$\mapsto$	$\{ex$	$\mapsto$	$\{\}\}$		
s-writing	$\mapsto$	$\{ex$	$\mapsto$	$\{question$	$\mapsto$	$r-,$
				$answer$	$\mapsto$	$rw\ \}\}$
s-waiting	$\mapsto$	$\{ex$	$\mapsto$	$\{question$	$\mapsto$	$r-,$
				$answer$	$\mapsto$	$r-\ \}\}$
s-done	$\mapsto$	$\{ex$	$\mapsto$	$\{question$	$\mapsto$	$r-,$
				$answer$	$\mapsto$	$r-,$
				$grade$	$\mapsto$	$r-\ \}\}$

Figure 4.4: Exam Workflow for one student and one teacher - Student's view

$$
\begin{array}{llllll}
\textit{t-ready} & \mapsto & \{ex & \mapsto & \{question & \mapsto & rw\}\} \\
\textit{t-waiting} & \mapsto & \{ex & \mapsto & \{question & \mapsto & r-\}\} \\
\textit{t-grading} & \mapsto & \{ex & \mapsto & \{question & \mapsto & r-, \\
 & & & & answer & \mapsto & r-, \\
 & & & & grade & \mapsto & rw\}\} \\
\textit{t-done} & \mapsto & \{ex & \mapsto & \{question & \mapsto & r-, \\
 & & & & answer & \mapsto & r-, \\
 & & & & grade & \mapsto & r-\}\}
\end{array}
$$

Figure 4.5: Exam Workflow for one student and one teacher - Teacher's view

4.1.2 Use case : Multiple Students & A Teacher

Now consider the more common use case involving n students taking an exam. The teacher distributes n exam papers to the n students, waits for all of them to complete and submit their answers, and then returns them to each student with a grade.

4.1.2.1 Process Layer

Figure 4.6 specifies the workflow for the n-student, one teacher exam scenario. At any given time, the ith student is in one of the following possible states: ready to take the exam *s_i-ready*, writing the exam *s_i-writing*, waiting for the grade *s_i-waiting*, and done, *s_i-done*. The teacher is in one of the following states: ready with the exam, *t-ready*, distributing the exam to the students, *t-dist*, waiting for all the students to submit the exam, *t-waiting*, grading, *t-grading*, and *t-done*. The state *t-dist* is parametrized by the parameter 'A' that denotes the set of students who have already received the question paper, *t-waiting* by the set of papers already submitted, and *t-grading* by the set of papers already graded.

The teacher starts in the state, *t-ready*, transitions to the state *t-dist* with an empty parameter list, sends the paper to each student (in any order) on the channel, and reaches the state *t-dist* parametrized with a subset A of the set {1,2...n} abbreviated N. On the distribution of the papers, the teacher transitions into the *t-waiting* state again with an empty parameter list. On receiving the submissions from all students on the channel (again in any order), it moves to the *t-waiting* parametrized by the set {1,2...n}. Next, on receiving all the grades it transitions to the *t-grading* state again with no parameters. All the grades are then sent out to the students (in any order), and on reaching the *t-grading* state with the parameter as the set {1,2...n}, the teacher finally transitions into the *t-done* state and exits. We can see that because of any number of random orderings possible, it is very difficult to represent the flow of states in a single graph.

In Figure 4.7, we make an attempt to show the graph for the workflow of a teacher in this case, where there are n students taking the exam. The graph for a student in this case isn't shown, as it is similar to the one in Figure 4.2. The nodes of the graph shown refer to the parametrized states. This graph

$$
\begin{array}{lcl}
s_i\text{-}ready & \stackrel{\text{def}}{=} & paper_i.\ s_i\text{-}writing \\
s_i\text{-}writing & \stackrel{\text{def}}{=} & \overline{submit_i}.\ s_i\text{-}waiting \\
s_i\text{-}waiting & \stackrel{\text{def}}{=} & grade_i.\ s_i\text{-}done \\
 & & \\
t\text{-}ready & \stackrel{\text{def}}{=} & \tau.\ t\text{-}dist(\{\}) \\
t\text{-}dist(A) & \stackrel{\text{def}}{=} & \sum_i \overline{paper_i}.\ t\text{-}dist(A \cup \{i\}) \\
t\text{-}dist(N) & \stackrel{\text{def}}{=} & \tau.\ t\text{-}waiting(\{\}) \\
t\text{-}waiting(A) & \stackrel{\text{def}}{=} & \sum_i submit_i.\ t\text{-}waiting(A \cup \{i\}) \\
t\text{-}waiting(N) & \stackrel{\text{def}}{=} & \tau.\ t\text{-}grading(\{\}) \\
t\text{-}grading(A) & \stackrel{\text{def}}{=} & \sum_i \overline{grade_i}.\ t\text{-}grading(A \cup \{i\}) \\
t\text{-}grading(N) & \stackrel{\text{def}}{=} & \tau.\ t\text{-}done
\end{array}
$$

Initial Process Expression:

$s_1\text{-}ready \mid \ldots \mid s_n\text{-}ready \mid t\text{-}ready$

Figure 4.6: Exam workflow: n Students & one Teacher

assumes that the ordering in which the exam is distributed, collected and graded is {1,2...n}. However, this is only one case, we can calculate that there will be O(n!) such cases.

4.1.2.2 Data Layer

The data model can be seen in Figure 4.8. There are three fields: *question*, *answer* and *grade*. There is just one form: the *exam* form.

4.1.2.3 View layer

Now we define the view layer for this example. The view associates user states with form view. Figures 4.9 and 4.10 capture the state-based access control for the student and teacher respectively.

4.1.3 Discussion on the exam workflow example

With this example, we can see how one can model and algebraically specify workflows which in this case was an exam workflow. This was a simple example, but such a method of formalization can be extended to large educational process as well. Intuitively, from this example it seems that all possible educational processes can be expressed by these ideas of ”flow of work“.

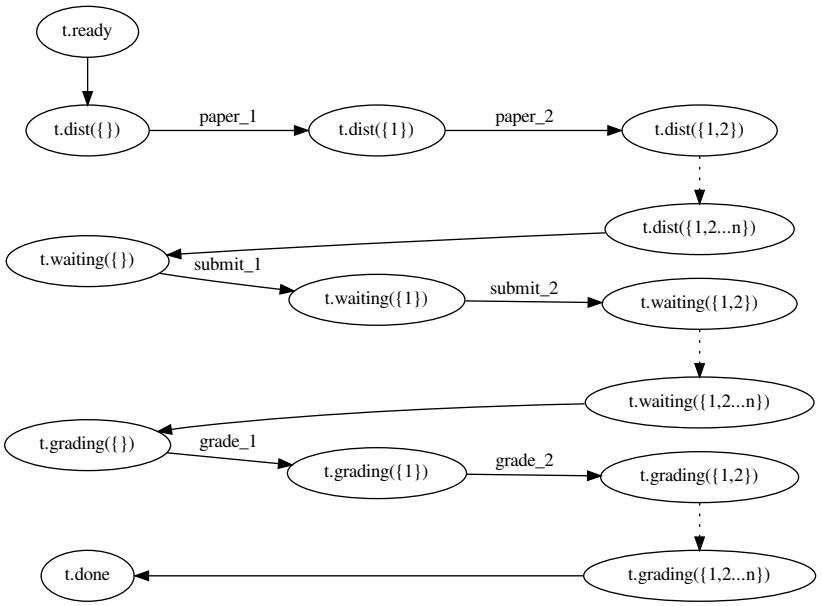


Figure 4.7: An attempt at constructing a Graph for a teacher participating in a n-student, one-teacher exam workflow

We also observe that graphical notation of modelling can't handle complex workflows, in turn proving the power of an algebraic notation such as π-calculus.

We further see that any system built on this algebraic specification will essentially be just an implementation which follows this algebraically specified workflow, taking action based on the current state and the information on the communication channel. This is a big advantage of expressing workflows in an algebraic way. It also allows such specifications to be used to statically and dynamically validate the protocol the system is following.

$$
\begin{array}{lcll}
\textit{users} & = & \{\sum_i s_i, t\} & \\
\textit{forms} & = & \{ex\} & \text{Exam Form} \\
\textit{fields} & = & \{\sum_i question_i, & \text{Questions} \\
 & & \sum_i answer_i, & \text{Answers} \\
 & & \sum_i grade_i\} & \text{Grades}
\end{array}
$$

Figure 4.8: Exam Workflow for n student and one teacher - Data Layer

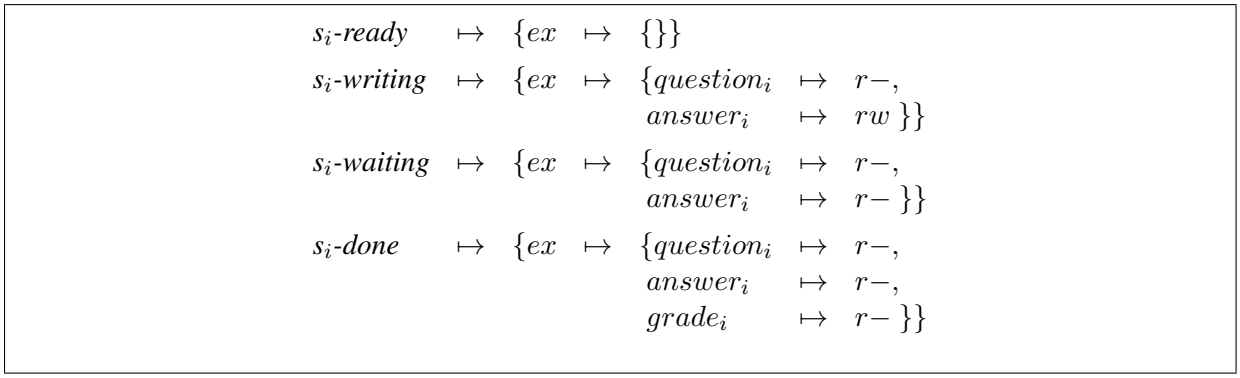


Figure 4.9: Exam Workflow for n students and one teacher - A student's view

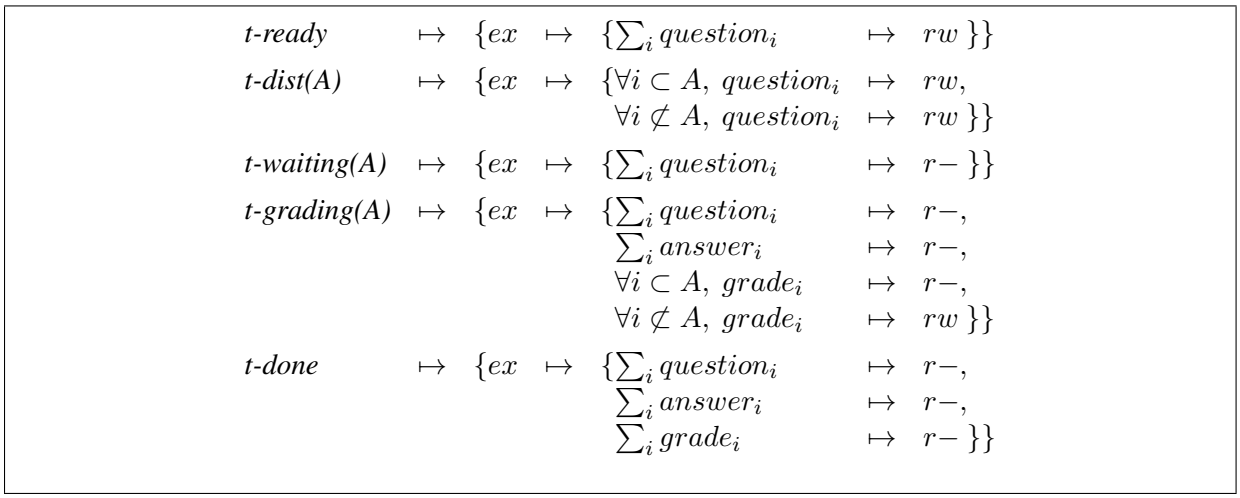


Figure 4.10: Exam Workflow for n student and one teacher - Teacher's view

4.2 Examples from E-governance

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 [57]. The primary recommendation is for all government processes to be re-engineered to suit computerization starting with about ten to twenty 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.

4.2.1 Passport Application process

Figure 4.11 presents a considerably simplified version of the Passport application process in India [35].

1. The applicant (citizen) submits a form with personal details to the Passport Office (PPO).
2. PPO receives the form. Then,
   (a) If the form is incomplete, the PPO sends back the form to the applicant
   (b) If the form is complete and meets the eligibility criteria, PPO forwards the application to the police department for verification.
3. Police receives the application form, and takes action 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 4.11: Passport application: Plain text specification

There are several fields in the form to be submitted in the application form for the passport. However, for sake of simplicity, we have considered only three fields: Name, Date of Birth(DOB), and Address. Each of the three layers of the model are described next.

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

Initial Process Expression:

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

Figure 4.12: Passport application workflow

4.2.1.1 Process Layer

Figure 4.12 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 has the following 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 4.13 maps the plain text specification to the CCS based workflow specification.

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 4.11. 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

Plain-text	Workflow Steps	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 4.13: Passport Application - Mapping of the plain text specification to the workflow specification

required. Similarly the police has just one action which is to confirm the application to the PPO, shown in Step 9 of the workflow specification.

4.2.1.2 Data Layer

The data model for this layer is given in Figure 4.14. 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 the 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.

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

Figure 4.14: Passport Application - Data Layer

4.2.1.3 View layer

The view connects the data and the process layers. The view layer associates user state with the form view. Figure 4.15 shows 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.

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

Figure 4.15: Passport Application - Citizen's view

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

Figure 4.16: Passport Application - PPO's view

Figures 4.16 and 4.17 are the views for the PPO and the police 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.

4.2.2 Public Grievance portal

The public grievance portal of India [34] 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

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

Figure 4.17: Passport Application - Police's view

government department against which the grievance is raised. In this process the identity of the citizen is hidden from the concerned government department. Figure 4.18 gives the specification of this process in plain text.

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 can:
   (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 returns the response back to the government department. Back to step 2.
   (b) If the response is satisfactory, PGO accepts the response and also forwards it to the citizen.

Figure 4.18: Grievance Portal - Plain text specification

4.2.2.1 Process Layer

Figure 4.19 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, or finally done *pgo-done*. *gov* can be in one of the following states: ready for processing the grievance *gov-ready*, addressing the grievance *pol-addressing*, waiting for the *pgo gov-waiting* and done *gov-done*. Figure 4.20 shows the map from the workflow specification map to the plain text specification.

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

Initial Process Expression:

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

Figure 4.19: Public Grievance Portal workflow

4.2.2.2 Data Layer

The data model can be seen in Figure 4.21. There are four fields: *name*, *address*, *grievance* and *response*. There are two forms: *grievance* form and *status* form.

4.2.2.3 View layer

Now we define the view layer for this example. The view associates user states with form view. Figures 4.22-4.24 capture the state-based access control for the citizen, PGO and the government department respectively.

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

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

Figure 4.21: Grievance Portal - Data Layer

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

Figure 4.22: Grievance Portal - Citizen's view

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

Figure 4.23: Grievance Portal - PGO's view

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

Figure 4.24: Grievance Portal - Government's view

The citizen uses the *grievance* form to submit a grievance, while the *status* form is used to collect the response from the *gov* and show it to the citizen. It can be seen that only 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 4.24, where the view of the government department is defined such that the department does not have access to the 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.

Chapter 5

Implementation as a Web application

The proposed model has been inspired by the idea of separating business processes from the user information [11]. Our model is layered in a way that, the data layer stores the user information, the process layer defines the business process, and the view layer handles the access control to the data. It has been designed keeping in mind that the end product will be a web application where different users can log in individually, and take part in the process.

This chapter discusses a prototype web-based implementation of the model. This is available publicly on GitHub [14]. For gathering the specification, we capture all three layers of the model separately from the user, i.e. the data layer that consists of just the different fields, forms and users; the process layer which is captured as a Pi-calculus specification and the view layer which is captured as updates based on the changes in the process layer.

A workflow engine is a key part of such web applications. The role of the workflow engine is to execute the workflow as per the specification. All commercially available WFMS have an inbuilt workflow engine, however none of them provide a support for formal algebraic specifications. For a comparison, we shall look at some of the available Pi-calculus engines in the next section. Apart from the workflow engine, there is also a need for a data store for storing the elements of the data layer.

The next section discusses some of the different implementations of Pi-calculus engines that are currently available.

5.1 Pi-calculus engine implementations

Turner, in his PhD thesis [71] introduces a basic implementation for Pi calculus called Pict. In this implementation, the calculus is polyadic, asynchronous, typed and also does not include summation. Turner also provides an implementation in C as a part of his thesis. The implementation uses FIFO queues for representing queues, and the data consists of boolean and integer values. The processes are represented as heap-allocated closures. A closure stores the code pointer (address to the function) along with the free variables associated with the process. Turner left out a distributed implementation as a part of his thesis.

PiLib [25] is also an implementation of Pi -calculus in Scala from Cremet and Odersky. It is shipped as a part of Scala's standard library. It uses a monadic, synchronous and typed version of the calculus and includes guarded summation. PiStache [53] is a recent implementation in Scala with some performance improvements.

The occam-pi language [6] is a variation of the original occam language built on the top of Pi-calculus, instead of the Communicating Sequential Processes (CSP). occam-pi implements a typed, synchronous and monadic variant of the calculus. Guarded sums and distributed process mobility are included as a part of the calculus. KRoC [6] is the compiler written on using occam-pi.

The JPie Interface [83] is a Java implementation made for grid computing. It is a set of java classes that implement the essential semantics of Pi-calculus. It allows for implementation as distributed processing systems in java.

We discussed some implementations of Pi-calculus in this Section. The next section discusses our implementation of Pi-calculus, which is core to our approach of a WFMS with access control.

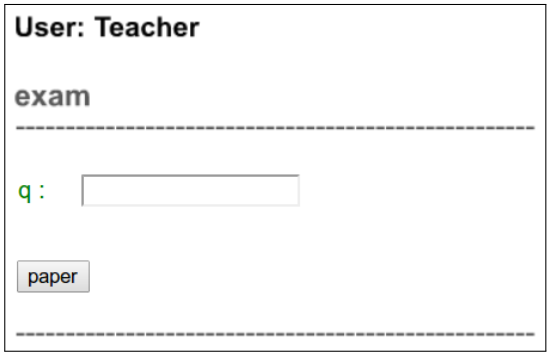
5.2 Implementation

We use the Python programming language [8] to implement our engine. We implement a polyadic, asynchronous, untyped version of Pi-calculus and also include summations. We also use queues similar to Pict to implement the queues, however we include support for summations. The view layer that control access control is closely tied with the workflow engine. We use MongoDB [5] as our data store, MongoDB is a NoSQL database that allows for the user process data to be serialized as an object, and can be saved and retrieved anytime directly from the database.

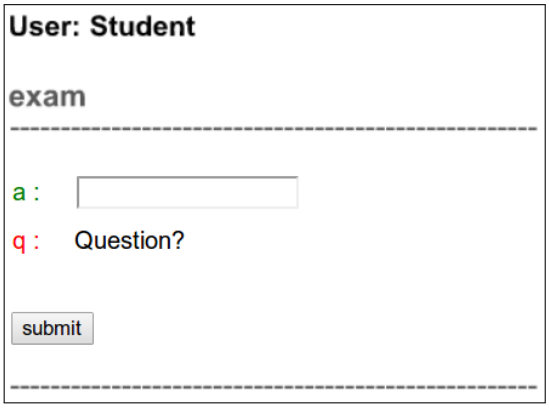
The web interface makes use of Modpython [4] in order to input the process specification, and also to generate the web application. The reader should note that the web interface uses the engine in order to process the user data, view and actions. We describe the functioning of the web interface below.

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 displayed 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.

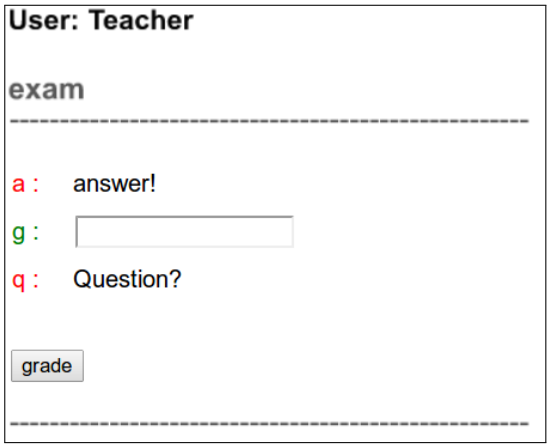
The next section demonstrates the implementation using the examples discussed in the previous section. We attempt to give the reader an understanding of the generated application with the help of screenshots. We show the fields with read write access in green colour, read only access in red colour, and write only access in blue colour.



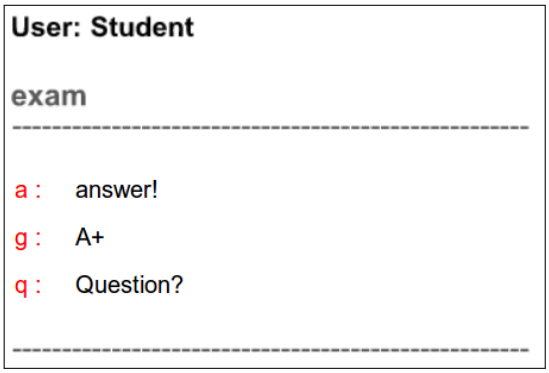
(a) Form for the teacher in t-ready state



(b) Form for the student in s-writing state



(c) Form for the teacher in t-grading state



(d) Form for the student in s-done state

Figure 5.1: Screenshots of the Exam workflow example.

5.3 Examples

5.3.1 Exam workflow

Firstly we discuss the simple exam workflow discussed in Section **??**. Figure 5.1 shows screenshots of the web application. Figure 5.1(a) shows the *t-ready* state, when the teacher is preparing the exam for the student. This is an form with read and write access to only the *q* field. Figure 5.1(b) shows the *s-writing* state. In this state the student has read-only access to the *q* field, and read-write access to the *a* field, and can submit the exam by clicking "submit". Similarly, Figure 5.1(c),(d) show the view of the users as the workflow progresses.

It should be seen that the users: *student* and *teacher* need to interact with each other in order for the workflow to proceed forward. Each user can be thought of as Deterministic Finite Automatons(DFA), the system evolves as a result of these different DFA's interacting with each other.

5.3.2 Passport Application Portal

We use the passport application example discussed in Section 4.2.1 to introduce the implementation. Figure 5.2 shows screenshots of the web application. Figure 5.2(a) shows the *c-ready* state, when the citizen is filling personal details for the application. This is an form with read and write access to the *name*, *dob* and *add* fields. This has been defined as a function of the state in the example. Figure 5.2(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 "verify". These buttons are based on the different state transitions available to the user from the current state. Similarly, Figure 5.2(c),(d) and (e) show the view of the users as the workflow progresses.

It is important to observe that the implementation acts like an interpreter of the workflow specifications, and thus building this example did not require any code to be written. This has many benefits over MVC models like Ruby on Rails, Django etc. Firstly, as only the specification needs to be written, the time taken to build an application is drastically reduced. Secondly, as there is no need for writing code, the need for testing is eliminated. There are some weaknesses of this application, but they are discussed later. The next subsection discussed the public grievance portal example.

5.3.3 Public Grievance Portal

Secondly we discuss the implementation of the Public Grienvance Portal example discussed in Section 4.2.2. Figure 5.3 shows screenshots of the web application as viewed by the different users as a part of their workflow. Figure 5.3(a) shows the *c-ready* state, when the citizen is filling the grievnace form. At this state, the citizen has read and write access to the *name*, *add* and *grievance* fields. Figure 5.3(b) shows the *pgo-reviewing* state. In this state the PGO has read-only access to the fields filled by the citizen, and can either decline the application back by clicking "decline" or send it to the government by clicking "req".

Figure 5.3(c) shows the form *r* for the government, when it has read only access to the grievance, and read write access to the response. Figure 5.3(d) shows the form *g* after it is verified by the police, and sent back to the PGO. Finally figure 5.3 shows the form *r* at *c-done* state for the citizen, where it gets read only access to the response from the government.

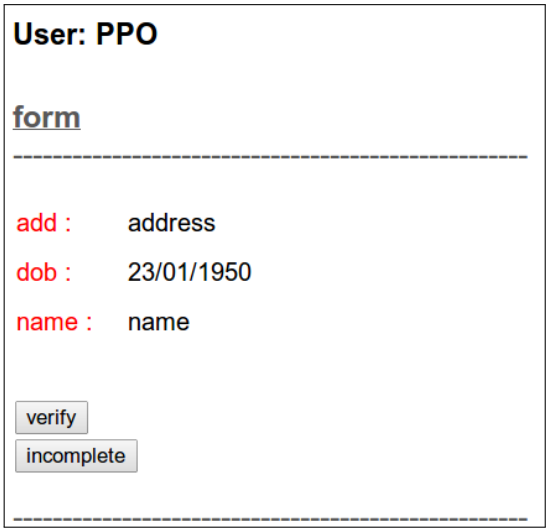
5.4 Discussion

In this chapter, we demonstrate a web application, which is build on top of a simple Pi-calculus engine. It consists of three layers: Process layer, implemented as a Pi-calculus engine; Data Layer,

implemented by the MongoDB database; View Layer, implemented by logic defined in the web server. The view of the user is controlled by the view layer, however any action taken by the user is handled by the engine, which executes the corresponding Pi-calculus action, that finally updates the view of the user.

Our approach of linking the view of the user with the pi-calculus engine via access control is a novel one. It however poses some limitations. Firstly, the data that was entered by the user, can't have any influence on the flow. For example, in the passport workflow example, there can not be a separate workflow for a child based on the age entered by the user. Secondly, our Pi-calculus implementation is not designed for distributed systems.

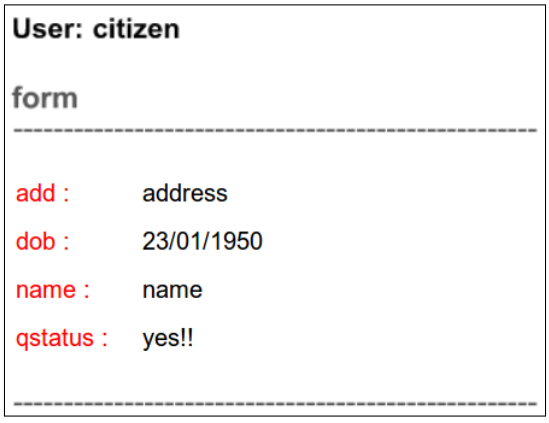
(a) Form for the citizen in c-ready state



(b) Form for the PPO in ppo-reviewing state

(c) Form for the police in pol-verifying state

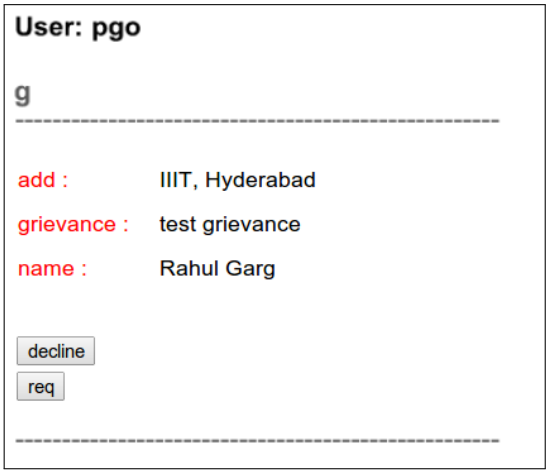
(d) Form for the PPO in ppo-verifying state



(e) Form the citizen in c-done state

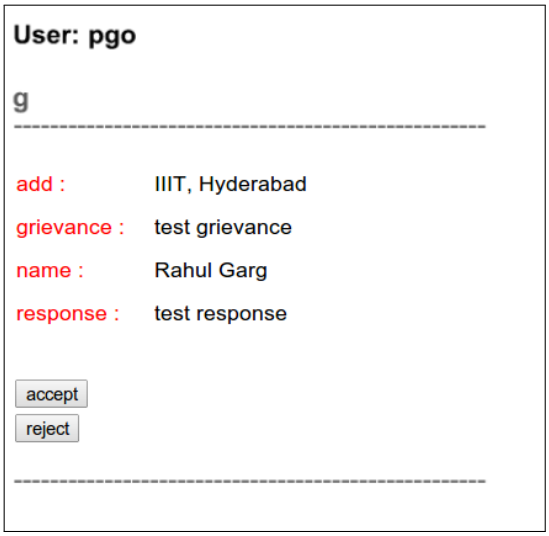
Figure 5.2: Screenshots of the public passport portal example.

(a) Form g for the citizen in c-ready state

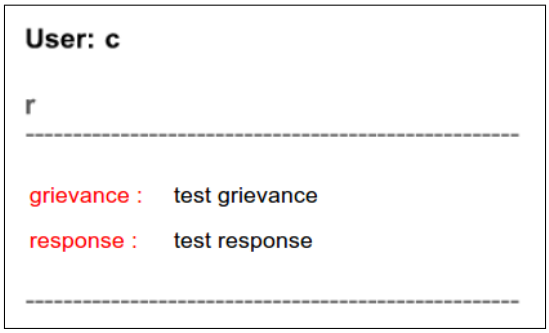


(b) Form g for the PGO in pgo-reviewing state

(c) Form r for the government in gov-addressing state



(d) Form g for the PGO in pgo-evaluating state



(e) Form r the citizen in c-done state

Figure 5.3: Screenshots of the public grievance portal example.

Chapter 6

A Formal model for expressing permissions using Pi-calculus

6.1 Introduction

The model discussed in the previous chapters uses Pi-calculus to formally model processes, while access control is modelled informally using handlers and updates. This chapter discusses a formal model based on Pi-calculus to express permissions over data for a process.

Milner [55] introduced the example of a buffer to model data using Pi -calculus. This example thinks of data as a process, in which the data is stored as a parameter to the process state. In this section, we initially extend this basic example to model permissions for a single buffer; and later propose a solution to model access control for a complete system.

6.2 Milner's model for a buffer

Milner proposes a model which takes a buffer as a process, which can be in one of the following states: *B* when its empty, and *C* when it has some data. This data is stored as a parameter *x* to the state *C*. Both the states are also parametrized on two channels, *r* for reading & *w* for writing.

$$B(r,w) \stackrel{\text{def}}{=} w(x).C(x,r,w)$$
$$C(x,r,w) \stackrel{\text{def}}{=} \overline{r}(x).B(r,w)$$

One can observe that since this model is only meant for a buffer, it has some shortcomings to be used to model a data cell. For example, after a read is performed, the data is lost from the buffer. Also, a user can't overwrite data. Thus, we extend this model of a buffer to consider persistent storage of the data.

$$B(r,w) \stackrel{\text{def}}{=} w(x).C(x,r,w)$$
$$C(x,r,w) \stackrel{\text{def}}{=} \overline{r}(x).C(x,r,w) + w(y).C(y,r,w)$$

We call this model a cell. Upon a read the data is not lost, and also data can be written even when the user is at state *C*. Note that, we can simplify this model to a single state if there is some initialization value for the data.

$$C(x, r, w) \stackrel{\text{def}}{=} \overline{r}(x).C(x, r, w) + w(y).C(y, r, w)$$

Suppose the initialization value is '0', then the initialization expression for this will be *C(0,r,w)* with *r* & *w* as the read and write channels provided to the user.

6.3 Model for a secure cell for a single user

$$\begin{aligned}
\textit{User}(us) \quad &\stackrel{\text{def}}{=} \quad \overline{us}.\ us(r,w).\\
&\quad (\overline{r}.\ r(v).\ \overline{\textit{Display}}\langle v\rangle \ +\ \textit{Get}(v).\ \overline{w}\langle v\rangle\ ).\\
&\quad \textit{User}(us)\\[1em]
\textit{System}(us) \quad &\stackrel{\text{def}}{=} \quad (us.(\ (new\ pch)\ \textit{Perm}(pch).\\
&\qquad \overline{pch}.pch(r,w).\overline{us}\langle r,w\rangle.pch\ )\ )\\
&\quad .\ \textit{System}(us)\\[1em]
\textit{Data}(v,rw,ro,\\
\quad ow,oo) \quad &\stackrel{\text{def}}{=} \quad (\ rw.\ ((new\ r\ w).\ \overline{rw}\langle r,w\rangle.\\
&\qquad (r.\ \overline{r}\langle v\rangle.\ \textit{Data}(v,rw,ro,ow,oo) +\\
&\qquad\quad w(d).\ \textit{Data}(d,rw,ro,ow,oo)\ ))\\
&\qquad + ro.\ ((new\ r\ w).\ \overline{ro}\langle r,w\rangle.\\
&\qquad (r.\ \overline{r}\langle v\rangle.\ \textit{Data}(v,rw,ro,ow,oo) +\\
&\qquad\quad w(d).\ \textit{Data}(v,rw,ro,ow,oo)\ ))\\
&\qquad + ow.\ ((new\ r\ w).\ \overline{ow}\langle r,w\rangle.\\
&\qquad (r.\ \overline{r}\langle 0\rangle.\ \textit{Data}(v,rw,ro,ow,oo) +\\
&\qquad\quad w(d).\ \textit{Data}(d,rw,ro,ow,oo)\ ))\\
&\qquad + oo.\ ((new\ r\ w).\ \overline{oo}\langle r,w\rangle.\\
&\qquad (r.\ \overline{r}\langle 0\rangle.\ \textit{Data}(v,rw,ro,ow,oo) +\\
&\qquad\quad w(d).\ \textit{Data}(v,rw,ro,ow,oo)\ )))
\end{aligned}$$

Initial Process Expression:

$$\textit{Data}(0, rw, ro, ow, oo) \mid \textit{System}(us) \mid \textit{User}(us)$$

Figure 6.1: Modelling permissions for a user on a field

In the previous model, it can be seen that if any user gets access to channels *r* & *w*, the user can get permanent read and write access to the cell. In order to model a real world system, we need to address this issue of access control. In this section, we propose a model which controls the access of a cell with the help of a third party, the system. The system acts as a layer between the user and the data process.

$$
\begin{array}{lcl}
\textit{User}_i(\textit{us}) & \stackrel{\text{def}}{=} & (new\ f).\ Get(f).\ \overline{us}\langle f\rangle.\ \textit{us(r,w)}. \\
\quad \forall i \in [1,n] & & (\bar{r}.\ \textit{r(v)}.\ \overline{\textit{Display}}\langle v\rangle\ +\ Get(v).\ \overline{w}\langle v\rangle\ ). \\
 & & \textit{User}_i(us) \\
 & & \\
\textit{System}(us_i & \stackrel{\text{def}}{=} & (\sum_{i=1}^{n} us_i(j).(\ (new\ pch)\ \overline{\textit{Perm}}\langle i,j\rangle.\ \textit{Perm(pch)}. \\
\quad \forall i \in [1,n]) & & \qquad \overline{pch}.\ \textit{pch(r,w)}.\overline{us_i}\langle r,w\rangle.\textit{pch}\ )\ ) \\
 & & .\ \textit{System}(us_i\ \forall i \in [1,n]) \\
 & & \\
\textit{Data}_j\textit{(v,rw,ro,} & \stackrel{\text{def}}{=} & (\ rw.\ (\textit{(new r w)}.\ \overline{rw}\langle r,w\rangle. \\
\quad \textit{ow,oo)} & & \quad (r.\ \bar{r}\langle v\rangle.\ \textit{Data}_j\textit{(v,rw,ro,ow,oo)} + \\
\quad \forall j \in [1,m] & & \qquad w\textit{(d)}.\ \textit{Data}_j\textit{(d,rw,ro,ow,oo)}\ )) \\
 & & + ro.\ (\textit{(new r w)}.\ \overline{ro}\langle r,w\rangle. \\
 & & \quad (r.\ \bar{r}\langle v\rangle.\ \textit{Data}_j\textit{(v,rw,ro,ow,oo)} + \\
 & & \qquad w\textit{(d)}.\ \textit{Data}_j\textit{(v,rw,ro,ow,oo)}\ )) \\
 & & + ow.\ (\textit{(new r w)}.\ \overline{ow}\langle r,w\rangle. \\
 & & \quad (r.\ \bar{r}\langle 0\rangle.\ \textit{Data}_j\textit{(v,rw,ro,ow,oo)} + \\
 & & \qquad w\textit{(d)}.\ \textit{Data}_j\textit{(d,rw,ro,ow,oo)}\ )) \\
 & & + oo.\ (\textit{(new r w)}.\ \overline{oo}\langle r,w\rangle. \\
 & & \quad (r.\ \bar{r}\langle 0\rangle.\ \textit{Data}_j\textit{(v,rw,ro,ow,oo)} + \\
 & & \qquad w\textit{(d)}.\ \textit{Data}_j\textit{(v,rw,ro,ow,oo)}\ )))
\end{array}
$$

Initial Process Expression:

$$
\begin{array}{c}
\textit{Data}_1(0)\ |\ \textit{Data}_2(0)\ |\ \ldots\ |\ \textit{Data}_m(0)\ | \\
\textit{System}(us_i\ \forall i \in [1,n])\ | \\
\textit{User}_1(us_1)\ |\ \textit{User}_2(us_2)\ |\ \ldots\ |\ \textit{User}_n(us_n)
\end{array}
$$

Figure 6.2: Modelling permissions for a system

This model consists of three processes: User, System and sData. User is the user process, the sData is a process that encapsulates the data, and System is the system process that controls the access of the user process. There are two channels: 'us', shared by User and System; and 'sc', shared by System and sData. Note that the user does not have a direct connection to sData, but can only access the data via System. System stores the permissions of the data for the user as a parameter to its state.

Now we discuss the high level functioning of the system. Whenever the user process wants to access the data, a request is sent over the channel sc. Upon receiving the request, the system based on the permission, queries sData, which returns back a new private channel which is sent back to User. User, can now send a request (read or write) on this channel, and sData responds. The scope of the private channel is limited to this request. Note that this private channel is created based on the permission of the user.

6.4 Model for a complete system with n users and m fields

The data model of a system consists of users and fields. This model extends the model defined in the previous section to incorporate n users and m fields.

Whenever a user 'u' queries for a field 'f', an interface of that field is returned to the user, based on the permission of the field at the current state, by using the "Perm" process. This process accepts the user accepting the field 'u', and field to be accessed 'f' as the arguments on the channel 'perm' and returns one of the four channels defined on f - 'rw', 'ro', 'ow' & 'oo' based on the access for that user. Note that in π-calculus, channels can be passed as parameters to a function.

6.5 Proof of security

The previous sections discuss the model in detail, however as the reader may remember, the key purpose of the model is to guarantee security. In this section, we first attempt to define security in the context of this model, and later prove it. The complexity comes from the fact that π-calculus allows passing of channels over channels. This is essential for the model, however, as a result proving security is a mammoth task due to the dynamic flow of channels.

We break the definition of security in two parts:

- *Isolation:* No two users processes can interact directly or share permissions with each other.
- *Access Control:* Users can read and write based on their access to the data defined by *Perm*

By proving *Isolation*, we can safely say that a user can not share or view data of another user process. *Access Control* proves that a user can read or write data based on her access to the data. Firstly we try to prove Isolation.

6.5.1 Isolation

We define η as a function, which for a given state, gives all the channels that can be accessed from that state, or any future reduction of that state.

Definition $\eta(u)$ = Set of channels accessible at state u or any β-reduction of u.

Definition Two users $User_i, User_j$ are set to be isolated iff $\eta(User_i) \cap \eta(User_j) = \varnothing$

We give the η for the different states.

$$\eta(User_i) = \{us_i,\ r,\ w\} \tag{6.1}$$

r, *w* are scoped channels, and generated for the scope of one request by the *Data* process. us_i is the channel used by $User_i$ to interact with the *System*. Next we look at the η for *System*.

$$\begin{aligned}\eta(System) = \{&us_i\ \forall i \in [1,n],\\ &rw_j,\ ro_j,\ ow_j,\ oo_j\ \forall j \in [1,m],\\ &r,\ w\}\end{aligned} \tag{6.2}$$

Similar to *User*, *r* and *w* are generated for the scope of one request by the *Data* process. Finally we look at the *Data* processes.

$$\eta(Data_j) = \{rw_j,\ ro_j,\ ow_j,\ oo_j,\ r,\ w\} \tag{6.3}$$

Lemma 6.5.1 $\eta(User_i) \cap \eta(User_j) = \varnothing$

Proof

$$\eta(User_i) = \{us_i,\ r,\ w\}$$
$$\eta(User_j) = \{us_j,\ r,\ w\}$$

Since *r* and *w* are scoped channels with the *Data* process, they are not shared in the above two states. Thus the intersection is an empty set.

Theorem 6.5.2 *$User_i$ and $User_j$, $\forall i \neq j$, can not communicate with each other on any channel.*

Proof This a simple extension of Lemma 6.5.1 which implies that two *User* processes do not share any channels, hence it is implicit that they can not directly communicate with each other.

Theorem 6.5.2 implies that two users can not interact directly with each other , and thus can not share permissions or data with each other.

6.5.2 Access Control

Proving access control maps down to proving the correctness of permissions. As there are a limited number of permissions, we prove this by simulating reads and writes over every permission channel for a data process. For reads, output on the channel *r* is the relevant data sent to the user, and in case of writes, the data value encapsulated by the data process after getting an input on channel *w* is the relevant data. Table 6.1 shows the output of the simulation. The state of the data process is *Data(v,rw,ro,ow,oo)*, and the value tried to be written is v_{new}.

We show the simulation for *rw* for the data process below. The rest were conducted similarly

$$
\begin{aligned}
\mathit{Data(v,rw,ro,oo,rw)} \quad &\stackrel{rw}{\Longrightarrow} \quad (r.\ \bar{r}\langle v\rangle.\ \mathit{Data(v,rw,ro,ow,oo)} + \\
&\qquad\qquad w(v_{new}).\ \mathit{Data}(w_{new},\mathit{rw,ro,ow,oo})\ )) \\
&\stackrel{r}{\Longrightarrow} \quad \bar{r}\langle v\rangle.\ \mathit{Data}_j\mathit{(v,rw,ro,ow,oo)} \\
&\stackrel{\bar{w}\langle v_{new}\rangle}{\Longrightarrow} \quad \mathit{Data}(v_{new},\mathit{rw,ro,ow,oo})
\end{aligned}
$$

Permission	Data read	Data after write
rw	v	v_{new}
ro	v	v
ow	*0*	v_{new}
oo	*0*	v

Table 6.1: Simulation output

Table 6.1 displays the output behavior for each permission on running simulations. This being clearly consistent with the permission model proves the *Access Control* property of security.

6.6 Discussion

In this chapter, we proposed two models for access control, first for a simple system with a user and a single field, and then for a complete system with n users and m fields. We take inspiration from Milner's model for implementing a cell using Pi-calculus. The view layer in our approach has the responsibility for controlling the access of a user. This is done by the help of a separate access control process which determines the access of a user for any field. We have also given a proof for security for the generic model.

Chapter 7

Conclusions

This thesis gave a state based access control approach to model processes as workflows. Our work aims at generating specifications for such systems. This specification can be verified and validated at run-time, using various analysis tools available for process algebra. We have also shown how such a specification can be used to generate a system with rules based on the specification. Chapter 5 gave the implementation for such a WFMS.

Chapter 3 gave a layered approach for access control in WFMS. It consisted of three layers: Data layer, View layer and Process Layer. The data layer is essentially the data store, which is used to save the user data, the process layer manages the workflow and the view layer is responsible for access control. Our approach uses state based access control, however we implement access control as logic in the part of the view layer, and not as a part of the π-calculus engine. In Section 6, we demonstrated modelling access control using π-calculus, which can also be used to further improve the approach, by allowing for algebraic verification techniques to be used to verify access control policies.

However the approach introduced in this thesis 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 [55] 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 workflows.

Related Publications

1. Ankur Goel, Venkatesh Choppella. Algebraic Modelling of Educational Workflows. In *IEEE* 4^{th} *International Conference on Technology for Education*, pages 153-156, IEEE Press, 2012.

2. Ankur Goel, Venkatesh Choppella. State Based Access Control for Open E-governance. In *Proceedings of the ACM International Conference on Theory and Practice of E-Governance (ICEGOV)*, ACM Press, 2013.

Bibliography

[1] Comindware Tracker. http://www.comindware.com/tracker/. [Online; accessed 2-Jan-2014].

[2] 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].

[3] IBM Lotus Workflow. www.ibm.com/lotus/workflow. [Online; accessed 2-Jan-2014].

[4] Modpython. http://www.modpython.org. [Online; accessed 2-Jan-2014].

[5] MongoDB. https://www.mongodb.org. [Online; accessed 2-Jan-2014].

[6] Occam-pi: blending the best of CSP and the pi-calculus. http://www.cs.kent.ac.uk/projects/ofa/kroc. [Online; accessed 2-Aug-2013].

[7] Process Maker. http://www.processmaker.com. [Online; accessed 2-Jan-2014].

[8] Python Programming Language. https://www.python.org. [Online; accessed 2-Jan-2014].

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

[10] Taverna. http://www.taverna.org.uk. [Online; accessed 2-Jan-2014].

[11] K. Aberer, A. Datta, Z. Despotovic, and A. Wombacher. Separating business process from user interaction in web-based information commerce. *Electronic Commerce Research*, 3(1-2):83–111, 2003.

[12] 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.

[13] G. J. Ahn, R. Sandhu, M. Kang, and J. Park. Injecting rbac to secure a web-based workflow system. In *Proceedings of the fifth ACM workshop on Role-based access control*, pages 1–10. ACM, 2000.

[14] Ankur Goel. Prototype Implementation using CCS. https://github.com/ankur1990/impl. [Online; accessed 2-Nov-2013].

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

[16] D. Beer, S. Höhne, H. Petersohn, T. Pöhnitzsch, G. Rünger, 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.

[17] 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.

[18] 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.

[19] E. Bertino, E. Ferrari, and V. Atluri. The specification and enforcement of authorization constraints in workflow management systems. *ACM Transactions on Information and System Security (TISSEC)*, 2(1):65–104, 1999.

[20] R. A. Botha. *CoSAWoE-a model for context-sensitive access control in workflow environments.* PhD thesis, 2008.

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

[22] R. A. Botha and J. H. P. Eloff. Separation of duties for access control enforcement in workflow environments. *IBM Systems Journal*, 40(3):666–682, 2001.

[23] A. D. Brucker, I. Hang, G. Lückemeyer, 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.

[24] 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.

[25] V. Cremet and M. Odersky. Pilib: A hosted language for pi-calculus style concurrency. In *Domain-Specific Program Generation*, pages 180–195. Springer, 2004.

[26] F. Cuppens and A. Miege. Administration model for or-bac. In *On The Move to Meaningful Internet Systems 2003: OTM 2003 Workshops*, pages 754–768. Springer, 2003.

[27] 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.

[28] D. E. Denning. A lattice model of secure information flow. *Communications of ACM*, 19(5):236–243, May 1976.

[29] R. Eshuis and J. Dehnert. Reactive petri nets for workflow modeling. In *Application and Theory of Petri Nets 2003*, pages 296–315. Springer, 2003.

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

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

[32] D. Georgakopoulos, M. Hornick, and A. Sheth. An overview of workflow management: from process modeling to workflow automation infrastructure. *Distributed and parallel Databases*, 3(2):119–153, 1995.

[33] C. K. Georgiadis, I. Mavridis, G. Pangalos, and R. K. Thomas. Flexible team-based access control using contexts. In *Proceedings of the sixth ACM symposium on Access control models and technologies*, pages 21–27. ACM, 2001.

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

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

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

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

[38] R. M. Hierons, K. Bogdanov, J. P. Bowen, R. Cleaveland, J. Derrick, J. Dick, M. Gheorghe, M. Harman, K. Kapoor, P. Krause, G. Lüttgen, A. J. H. Simons, S. Vilkomir, M. R. Woodward, and H. Zedan. Using formal specifications to support testing. *ACM Comput. Surv.*, 41(2):9:1–9:76, Feb. 2009.

[39] P. Jian, H. J. Hsu, and F. J. Wang. A delegation framework for access control in wfms based on tasks and roles. In *Future Trends of Distributed Computing Systems, 2008. FTDCS '08. 12th IEEE International Workshop on*, pages 165–171, 2008.

[40] 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].

[41] A. A. E. Kalam, R. Baida, P. Balbiani, S. Benferhat, F. Cuppens, Y. Deswarte, A. Miege, C. Saurel, and G. Trouessin. Organization based access control. In *Policies for Distributed Systems and Networks, 2003. Proceedings. POLICY 2003. IEEE 4th International Workshop on*, pages 120–131. IEEE, 2003.

[42] 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. Springer-Verlag.

[43] M. H. Kang, J. S. Park, and J. N. Froscher. Access control mechanisms for inter-organizational workflow. In *In Proceedings of the Sixth ACM Symposium on Access control models and technologies (2001), ACM*, pages 66–74. Press, 2001.

[44] K. Knorr. Dynamic access control through petri net workflows. In *Computer Security Applications, 2000. ACSAC'00. 16th Annual Conference*, pages 159–167. IEEE, 2000.

[45] K. Knorr. Multilevel security and information flow in petri net workflows. In *Proceedings of the 11th Conference on Advanced Information Systems Engineering*. Citeseer, 2001.

[46] K. Knorr. Www workflows based on petri nets. In *Contemporary Trends in Systems Development*, pages 331–343. Springer, 2001.

[47] K. Knorr and H. Weidner. Analyzing separation of duties in petri net workflows. In *Information Assurance in Computer Networks*, pages 102–114. Springer, 2001.

[48] G. E. Krasner, S. T. Pope, et al. A description of the model-view-controller user interface paradigm in the smalltalk-80 system. *Journal of object oriented programming*, 1(3):26–49, 1988.

[49] B. W. Lampson. Protection. *ACM SIGOPS Operating Systems Review*, 8(1):18–24, 1974.

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

[51] P. Lawrence. *Workflow handbook 1997*. John Wiley & Sons, Inc., 1997.

[52] T. Luo, L. Shi, F. Hong, et al. Perti net-based workflow access control model. *Journal of Shanghai University (English Edition)*, 8(1):63–69, 2004.

[53] P. Matiello and A. C. de Melo. Pistache: implementing π-calculus in scala. In *Formal Methods, Foundations and Applications*, pages 76–91. Springer, 2011.

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

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

[56] 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.

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

[58] U. Nations. Millennium development goals. `http://un.org/special-rep/ohrlls/lldc/MDGs.pdf`, 2000. [Online; accessed 2-Nov-2013].

[59] 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.

[60] 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.

[61] J. S. Park, G.-J. Ahn, and R. Sandhu. Role-based access control on the web using ldap. In *Database and Application Security XV*, pages 19–30. Springer, 2002.

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

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

[64] L. Qiu, Y. Zhang, F. Wang, M. Kyung, and H. R. Mahajan. Trusted computer system evaluation criteria. In *National Computer Security Center*, 1985.

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

[66] R. Sandhu. Lattice-based access control models. *Computer*, 26(11):9–19, 1993.

[67] R. S. Sandhu, E. J. Coyne, H. L. Feinstein, and C. E. Youman. Role-based access control models. *Computer*, 29(2):38–47, Feb. 1996.

[68] H. Smith and P. Fingar. Workflow is just a pi process. *BPTrends, November*, pages 1–36, 2003.

[69] Thomas, Sandhu, and R. K. Thomas. Task-based authorization controls (tbac): A family of models for active and enterprise-oriented authorization management. In *In Proceedings of the IFIP WG11.3 Workshop on Database Security, Lake Tahoe*, pages 166–181, 1997.

[70] R. K. Thomas. Team-based access control (tmac): a primitive for applying role-based access controls in collaborative environments. In *Proceedings of the second ACM workshop on Role-based access control*, RBAC '97, pages 13–19, New York, NY, USA, 1997. ACM.

[71] D. Turner et al. The polymorphic pi-calculus: Theory and implementation. 1996.

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

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

[74] J. Wainer, P. Barthelmess, and A. Kumar. W-rbaca workflow security model incorporating controlled overriding of constraints. *International Journal of Cooperative Information Systems*, 12(04):455–485, 2003.

[75] W. Wang, H. Ding, J. Dong, and C. Ren. A comparison of business process modeling methods. In *Service Operations and Logistics, and Informatics, 2006. SOLI'06. IEEE International Conference on*, pages 1136–1141. IEEE, 2006.

[76] 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.

[77] Wei-Kuang Huang and V. Atluri. A petri net based safety analysis of workflow authorization models. *Journal of Computer Security*, 8(2):209–240, 2000.

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

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

[80] C. Wolter, M. Menzel, and C. Meinel. Modelling security goals in business processes. In *Modellierung*, volume 127, pages 201–216, 2008.

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

[82] S. Wu, A. Sheth, J. Miller, and Z. Luo. Authorization and access control of application data in workflow systems. *Journal of Intelligent Information Systems*, 18(1):71–94, 2002.

[83] V. Yarmolenko, P. Cockshott, and E. Borland. Jpie interface: A java implementation of the pi-calculus for grid computing. 2004.

[84] 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.

[85] H. Zhao, Z. Fang, P. Xu, L. Zhao, J. Liu, and T. Wang. An improved role-based workflow access control model. In *Proceedings of the Fifth International Conference on Information Technology: New Generations*, ITNG '08, pages 551–556, Washington, DC, USA, 2008. IEEE Computer Society.