Towards Transparent E-Government Systems
– a View from Formal Methods

by

Xiaoyi CHEN

submitted to
Japan Advanced Institute of Science and Technology
in partial fulfillment of the requirements
for the degree of
Doctor of Philosophy

Supervisor: Professor Kokichi FUTATSUGI

School of Information Science
Japan Advanced Institute of Science and Technology

February 2009
Abstract

Government transparency should be embedded in the designs of e-Government systems. Lack of transparency can prevent the public from participating actively in government operating, such as raising questions and protesting ill-advised decisions, which in turn may cause concealing official graft or favoritism. However, it is also difficult to guarantee that designs of e-Government systems are consistent with real requirements in terms of having desired properties, such as transparency.

In this thesis, we pursue, in general, a better understanding and effective gauging method of government transparency through a specific combination of formal methods (the OTS/CafeOBJ method), software engineering and domain engineering’s views. Traditional methodologies in social sciences are normally judging the transparency and other properties of an e-government system afterward; we are proposing a method from software engineering side of view, to evaluate the transparency and other desired properties from the starting stage of a government project.

To achieve above goal, we carry out our work as follows:
Firstly, provides preliminary knowledge on Domain Methodology, CafeOBJ algebraic specifications, the OTS/CafeOBJ method, and its’ application example in e-Government system analysis. It focuses on introducing how to apply domain engineering methodology into real world system understanding and how the structural induction of the OTS/CafeOBJ method works for software design/specification verification of invariant properties by using the CafeOBJ system as an interactive theorem prover.
Secondly, presents the formal domain description based on the domain understanding firstly, and describes a General Service Delivery (GSD) model from public administration domain; and gives the CafeOBJ specification of it as well. The following transparency property discussion is based on this model.
Then, a literature overview around transparency is presented; the narrative definition of transparency is also refined. And based on the narrative description of transparency, we give the formal understanding and discussion of transparency based on the GSD model.
At last, we test our method on three government application cases and propose a public government license language (PGLL), which has potential possibility to improve transparency based on the previous formal discussion, and employ the OTS/CafeOBJ method to specify the PGLL as well.

Our method is not panacea for solving all government transparency problems for sure. However, we are providing a constructive suggestion from views of software engineering, formal method and domain engineering for supporting to realize a more transparent e-Government system.

Key Words: Transparent, The OTS/CafeOBJ method, E-Government, Public Administration, Domain Engineering
# Contents

Abstract i

1 Introduction 1
   1.1 Public Administration and E-Government 2
      1.1.1 Public Administration 2
      1.1.2 E-Government 3
      1.1.3 E-Government and the Public Administration Reforms 4
   1.2 Transparency in Current E-Government Practice 5
   1.3 Research Motivation 5
      1.3.1 Why the Domain Discussion 6
      1.3.2 Why the OTS/CafeOBJ Method 7
   1.4 Thesis Organization 7

2 Preliminaries 9
   2.1 Narrative Description of the Domain 9
      2.1.1 Introduction of Public Administration Domain 9
      2.1.2 Entities, Functions, Events and Behaviors 10
      2.1.3 Other Domain Aspects 13
      2.1.4 E-Government with Public Administration Domain 19
   2.2 The OTS/CafeOBJ Methods 19
      2.2.1 Observational Transition Systems 20
      2.2.2 Description of OTSs in CafeOBJ 20
      2.2.3 Analysis of OTSs 21
   2.3 Application Example of OTS/CafeOBJ Method in Government System Analysis 23
      2.3.1 Application Background 23
      2.3.2 A Messaging Framework 24
      2.3.3 Modeling and Specifications 25
      2.3.4 Analysis 28
   2.4 Summary 33

3 A General E-Government Model 35
   3.1 Formal Description of Public Administration Domain 35
      3.1.1 Entities, Functions, Events and Behaviors 35
      3.1.2 Simple Reasoning based on Domain Description 41
   3.2 Formal Description of General Service Delivery Model 42
      3.2.1 Introduction of Public Service Delivery 42
4 Study on Government Transparency

4.1 Literature Overview .............................................. 48
  4.1.1 Narrative Description of Transparency .................. 49
  4.1.2 Categories of Transparency ............................... 50
  4.1.3 A Brief Overview ........................................... 51
  4.1.4 Transparency with E-Government. ......................... 55
4.2 Transparency Evaluation Methodology ......................... 59
4.3 Formal Understanding of Transparency ......................... 59
  4.3.1 Formal Description of Transparency based on GSD Model .. 59
  4.3.2 Formal Description of Transparency based on GSD Model .. 60
  4.3.3 Formal Discussion of Transparency ....................... 61
4.4 Summary .......................................................... 62

5 Three Case Studies for Transparency Discussion ................. 63
5.1 Social Welfare .................................................... 63
  5.1.1 Narrative Description of Social Welfare .................. 63
  5.1.2 The OTS/CafeOBJ Specification of the Social Welfare Case 65
  5.1.3 Transparency Discussion of Social Welfare Case ........ 67
5.2 Business Licensing ............................................... 67
  5.2.1 Narrative Description of Business Licensing ............ 67
  5.2.2 The OTS/CafeOBJ Specifications of the Workflow ........ 68
  5.2.3 Transparency Discussion of Business Licensing Case ..... 71
5.3 Financial Budget Disclosure .................................... 72
  5.3.1 Narrative Description of Fiscal Disclosure ............... 72
  5.3.2 The OTS/CafeOBJ Specifications of the Workflow ........ 74
  5.3.3 Transparency Discussion of Financial Budget Disclosure .. 75
5.4 Simulation Discussion ............................................ 75
5.5 Summary .......................................................... 78

6 A Public Government License Language .......................... 79
6.1 Concept of License ............................................... 79
  6.1.1 A Concept of License ...................................... 79
  6.1.2 Concept of License Language ............................... 80
6.2 Public Government License Language ............................ 82
  6.2.1 Syntax Licenses ............................................. 82
  6.2.2 Annotations: License ...................................... 83
  6.2.3 Syntax: Actions ............................................ 85
  6.2.4 Annotations: Actions ...................................... 85
6.3 Experiments with PGLL ........................................... 86
  6.3.1 Experiments with Case Studies ............................. 86
  6.3.2 CafeOBJ Specification with PGLL ......................... 91
6.4 Summary .......................................................... 94
<table>
<thead>
<tr>
<th>Section</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>7 Conclusions</td>
<td>96</td>
</tr>
<tr>
<td>7.1 Summarization</td>
<td></td>
</tr>
<tr>
<td>7.2 Related Work</td>
<td>98</td>
</tr>
<tr>
<td>7.3 Future Work</td>
<td>100</td>
</tr>
<tr>
<td>A Glossary</td>
<td>101</td>
</tr>
<tr>
<td>B OTS/CafeOBJ Specifications of the Domain</td>
<td>106</td>
</tr>
<tr>
<td>C General Service Delivery (GSD) Model</td>
<td>112</td>
</tr>
<tr>
<td>C.1 Specification of GSD Model</td>
<td></td>
</tr>
<tr>
<td>C.2 Transparency Definition based on the GSD Model</td>
<td>121</td>
</tr>
<tr>
<td>D Case Studies with Transparency Discussion</td>
<td>123</td>
</tr>
<tr>
<td>D.1 OTS/CafeOBJ Specifications of Social Welfare Case</td>
<td></td>
</tr>
<tr>
<td>D.2 Refinement Proof</td>
<td>133</td>
</tr>
<tr>
<td>E Specification of PGLL</td>
<td>136</td>
</tr>
<tr>
<td>References</td>
<td>143</td>
</tr>
<tr>
<td>Publications</td>
<td>149</td>
</tr>
</tbody>
</table>
Chapter 1
Introduction

The rise of the information society has led to major changes in citizen expectations and organizational structures, cultures and working processes of public administration. Governments will have to adopt information society tools and working practices if they are to remain responsive to citizen needs. The Organization for Economic Co-operation and Development (OECD\(^1\)) defines e-Government as “the use of information and communication technologies, and particularly the Internet, as a tool to achieve better government”. The impact of e-Government at the broadest level is simply better government by enabling better policy outcomes, higher quality services, and greater engagement with citizens. Government and public administration will, and should, continue to be judged against these established criteria for success.

Lots of work has been done for pursuing the above criteria of “good government” in e-Government implementation, among which there are successful examples, such as the e-Macao project\(^2\). However, previous studies have also indicated many examples of e-Government failures. Heeks’ work \([64]\) stated that in developing countries, 35\% of e-Government projects are total failures, 50\% are partial failures, and only 15\% are successes. Therefore recently, it is commonly believed that how to avoid e-Government failure should be one of the important further studies in this area.

To our understanding, one critical reason of e-Government failure is that the informal criteria for success are difficult to be accounted. Considering the notion of transparency, which is an important aspect of the criteria: on one hand, although it has been studied broadly in various academic fields since 1990s (e.g. \([4]\), \([74]\)), it has been scarcely defined formally in a precise way; on the other hand, the informal descriptions of e-Government system designs actually prevent this criterion from being formally reasoned in a rigorous way.

We introduce formal methods into the field of e-Government for formalizing e-Government system designs and analyzing if the designs satisfy desired properties, in particular transparency that we focus on. Formal methods are mathematics-based approaches to software and hardware system development. Three major advantages of formal methods are: precise, concise, and allowing for proofs. Generally, “precise” means that the formal descriptions (called specifications) generated using formal methods are unambiguous and can facilitate the understanding and implementation by software developer and other stake-holders; “concise” means that the formal specifications capture all important issues

\(^1\)URL: http://www.oecd.org/
\(^2\)URL: http://www.emacao.gov.mo/
of a given system in a brief way; and “allowing for analysis” means that we can analyze desired properties of a given system based on its formal specifications.

Specifically, the formal method that we used is an equation-based method - the OTS/CafeOBJ method [62]. In using this method, an e-Government system design is first modeled as an OTS, a kind of transition system. The OTS is then specified in terms of equations using CafeOBJ, an algebraic specification language. And last, we express the transparency property as properties of the OTS, and verify the invariants by using the CafeOBJ system as an interactive theorem prover.

1.1 Public Administration and E-Government

In the first years of the twenty-first century, as the economies of many countries change rapidly with the information revolution, it is undeniable that there will be effects on the operations of government. Parallel to developments in the private sector in e-business and e-commerce, there have been sufficient governmental changes induced to warrant the name ‘e-government’ - a term becoming more used within public administration.

The public administration reforms and e-business, e-commerce and e-government are related reform movements. Bellamy and Taylor argue “The patterns of organizational change which are so commonly associated with the information age are remarkably consistent with the patterns associated with current forms of managerialism in public administration” [14]. Obviously the technological changes have had an impact on the public administration reforms; but rather than them being a technological determinant that inevitably led to new public management, it is rather the case that e-government merely reinforces the change to new forms of managing which were already occurring.

In a networked age, formal structures designed for nineteenth-century technology are unlikely to remain relevant. Information technology, particularly but not exclusively the use of computers, changes management; even the hierarchy itself. Managers do not need to wait until an item makes its way through the hierarchy; copies appear on their own computer screens. Information and data of all kinds can be gathered and transmitted cheaply, and transformed into performance information, which, in turn, allows management to be decentralized, even as it is able to be monitored by senior management. Records are kept electronically so that they are accessible from many different locations at the same time. Investments in information technology have resulted in less worker time spent in processing routine work.

Of course, the existence of a technology does not, by itself, determine the desired outcomes. There is no automatically change. The existence of technology does not, by itself, determine the predicted outcomes; it is rather that the technology makes changes possible. Therefore, in this research, we try to first understand the possible changes, and lead the changes into our desired way.

1.1.1 Public Administration

Public administration has a long history, one paralleling the very notion of government. As Gladden noted, some form of administration has existed ever since there have been governments: first comes the initiator or leader society possible, then the organizer or administrator to give it permanence. Administration, or the management of affairs, is the middle factor in all social activity, unspectacular but essential to its continuance.
The traditional model of public administration\textsuperscript{3} is also once a major reform movement. Where previously amateurs bound by personal loyalties to leaders carried out public administration, the task became a professional occupation which was carried out by a distinct merit-based public service. Serving the public at that time was a high calling, one that required the best people available to form distinct administrative elite and to act always according to the law and established precedents. Politicians might come and go but, while the apparatus of government was in the hands of permanent officials, the transition between regimes could be handled smoothly. Public administration as both theory and practice began in the late nineteenth century became formalized somewhere between 1900 and 1920, and lasted in most western countries largely unchanged until the last quarter of the twentieth century. This is a long period for any social theory, even if, since the early 1980s, governments have moved away from many of its precepts.

The traditional model can be characterized as: an administration under the formal control of the political leadership, based on a strictly hierarchical model of bureaucracy, staffed by permanent, neutral and anonymous officials, motivated only by the public interest, serving any governing party equally, and not contributing to policy but merely administering those policies decided by the politicians. Its theoretical foundations mainly derive from Woodrow Wilson and Frederick Taylor in the United States, Max Weber in Germany, and the Northcote-Trevelyan Report of 1854 in the United Kingdom.

The traditional model of public administration remains the longest standing and most successful theory of management in public sector, but is now being replaced. It has not disappeared overnight and elements of it still exist, but its theories and practices are now considered old-fashioned and no longer relevant to the needs of a rapidly changing society.

\subsection{1.1.2 E-Government}

E-Government\textsuperscript{4} refers to the use of Internet technology as a platform for exchanging information, providing services and transacting with citizens, businesses, and other arms of government. E-Government may be applied by the legislature, judiciary, or administration, in order to improve internal efficiency, the delivery of public services, or processes of democratic governance. The primary delivery models are Government-to-Citizen or Government-to-Customer (G2C), Government-to-Business (G2B) and Government-to-Government (G2G) and Government-to-Employees (G2E). Though e-government is often thought of as “online government” or “internet-based government,” many non-Internet “electronic government” technologies can be used in this context. Some non-internet forms include telephone, fax, and so on.

And there are different ways of classifying e-government interactions. In the development of e-business, the earliest phases are those where information only is provided and later there are two-way transactions. There are also different kinds of interaction possible between government and other parties, notably with citizens, businesses and other government agencies. There is a generally agreed four-stage set of developments dependent on the level of interaction allowed, particularly by websites.

\textsuperscript{3}Over the past 20 years the public sectors have undergone major change as governments try to respond to the challenges of technological change, globalization and international competitiveness. So the traditional model here is a kind of stable model which still influences most of countries.

\textsuperscript{4}from electronic government, also known as e-gov, digital government, online government or in a certain context transformational government.
(i) Information, the first stage, which is as far as most governments had progressed by 2002, involves departments and agencies using the World Wide Web to post information about them for the benefit of external users. (ii) Interaction, these sites become tools for two-way communication, allowing citizens to provide new information about themselves, gathered using instruments such as e-mail. (iii) Processing, a formal quantifiable exchange of value takes place, such as paying a license or a fine, even filing a tax return. It allows for tasks, previously carried out by public servants, to become web-based self-services, although they require off-line channels for completion. (iv) Transaction, this is where a portal for a wide range of government services is provided. But, e-Government much more than a simple web site. It is able to integrate government services and provide a path to them based on citizens’ needs, replacing the traditional structure of department or agency. Through introducing ICT\textsuperscript{5}, the information systems of all departments and agencies can be linked to deliver integrated services in a way that avoids users having to understand the agency structures of government.

1.1.3 E-Government and the Public Administration Reforms

As the e-government we described above, the changes induced to the operations of government are likely to be far-reaching. Public organizations have never been averse to using technology, but they have necessarily only been able to operate within the level of technology available. As that changes so must the way that government is organized and operates. As noted by The Economist in a special survey, “Within the next five years e-government will transform not only the way in which most public services are delivered, but also the fundamental relationship between government and citizen. After e-commerce and e-business, the next Internet revolution will be e-government”. The precise effects are difficult to predict, but that there will be major change driven by technology is a safe assumption to make.

In some respects, e-Government can be even considered a second managerial reform, another stage in the public management reforms that commenced in the 1980s. It does present a further challenge to the traditional model of public administration and, if implemented well, will transform the way public services are organized and delivered. E-Government may lead to changes in the political system as well as the internal operations of government. With the development of e-Government, terms such as “electronic democracy”, “digital democracy” are becoming more common. However, e-Government is a kind of ground-breaking study area. The traditional public administration study methodology can not fulfill all e-Government system requirements and discussion, because of the employment of information and communication technologies.

There are many considerations and potential implications of implementing and designing e-government, including disintermediation of the government and its citizens, impacts on economic, social, and political factors, and disturbances to the status quo in these areas.

So in this research, we are pursuing more new relationships which electronic technologies can benefit or bring reform chance on traditional public administration.

\textsuperscript{5}Information and Communication Technologies for short
1.2 Transparency in Current E-Government Practice

“E-Government has the potential to alter the traditional relationship between government and citizens by creating a new virtual government-and citizen interface” [79]. E-government, with ICT definition, there are common agreement that e-Government can enhance the transparency in traditional public administration [23, 1]. Although in different e-Government initiatives, different technologies are applied for achieving government performance evaluation, including accountability, participation, transparency and so on [23, 56, 36].

In paper [23], author applies a two stage multiple equation model and finds that internet use is positively associated with transparency satisfaction but negatively associated with interactivity satisfaction, and that both interactivity and transparency are positively associated with citizen trust in government. And transparency satisfaction in this work was viewed as the level of reliability of the information provided by government websites.

And the Cyberspace Public Research Group 2001’s, or CyPRG’s, Web Attribute Evaluation System (WAES) provides two broad dimensions (interactivity and transparency) for evaluating federal websites that could quite easily be modified for evaluating websites of any level of government (or even the private sector). Additionally, in his research on state websites. Assessing e-government: the Internet democracy, and service delivery6. West outlines a methodology for evaluation that leads to substantive conclusions regarding the status and success of implementation: government websites are not making full use of available technology, and there are problems in terms of access and democratic outreach. Taken as a whole these studies are indicative of a general tendency in the literature to define criteria for evaluation and implementation in general, but do not give insight into the current status of implementation of specific functions [12].

However, narrative description and performance evaluation afterward are difficult to instruct e-Government system implementation as well as reduce risk of e-Government project.

1.3 Research Motivation

Thus we take the property -transparency, which should be embedded in e-Government system from e-democracy, to explore the opportunities for the application of Formal Techniques to analyze and guarantee certain property in Electronic government and propose corresponding solution for it.

In this thesis, we want to reach a better understanding about e-Government systems and the how to embedded the desired property such as transparency in it. And our specific focus is on transparency improvement in e-Government systems from a formal methods point of view, namely that:

How the formal techniques can help to improve transparency property from the initial stage of e-Government system implementation and guarantee the desired properties (such as: transparency) be embedded into the systems.

The OTS/CafeOBJ method is a modeling, specification and verification method. In

---

6http://www.insidepolitics.org/govtreport00.html
the OTS/CafeOBJ method, a system to be verified is first modeled as an observational transition system (OTS), a transition system that can be straightforwardly written in terms of equations; The OTS is then written in CafeOBJ [9, 62], an algebraic specification language; Desired safety and liveness properties of the OTS can then be verified by using the CafeOBJ system as an interactive theorem prover. The main techniques used in this method to do verification include: case splitting, lemma discovery and structural induction.

To explore the opportunities for the application of Formal Techniques analyze and guarantee certain property in E-Government and propose corresponding solution for it. We want to achieve from the following parts:

- E-Government and public administration reforms: to understand the implications of e-Government for public administration; to explain the terminology which be involved in this research;
- Meanwhile, declare the formal definitions of related concepts: though many concepts such as transparency have been wide used, the definitions are ambiguous or difficult to be formalized as system requirements.
- Formal analysis of transparency: to analyze if the designs of e-Government systems satisfy transparency property, especially, by concrete case studies.
- Methodology to enhance transparency: from formal studies of the systems, we try to propose a language and discuss the potential usage in e-Government system.

Two questions that usually show up are (1) why we have domain discussion at first and; (2) why choose the OTS/CafeOBJ method. In the following subsections, we will try to explain the reasons and argue our position.

1.3.1 Why the Domain Discussion

The main purpose of introducing domain discussion is as following:

- Before software can be designed, programmed, coded, its requirements must first be reasonably well understood.
- Before requirements can be expressed properly, the domain of the application must first be reasonably well understood.

Therefore in order to conduct proper e-Government system development:

- One must first describe, informally and formally, the domain of the application.
- Then core parts of the requirements prescription must be somehow “derived” from the domain description.
- Finally the software design specification is somehow “derived” from the requirements prescription.

So for the formal specifications and discussions, the thesis presents the domain understanding firstly.
1.3.2 Why the OTS/CafeOBJ Method

The primary reason that we choose the OTS/CafeOBJ method is that: A number of already published case studies have shown the usability and the effectiveness of the OTS/CafeOBJ method, especially of its interactive inductive verification techniques for proving invariants with respect to OTS models. The published case studies of this method includes: [70, 44, 45, 43, 47, 39, 48, 46, 76, 75] etc, among which real world applications include some E-Commerce protocols, such as the SET electronic transaction protocol.

Additionally, some important features of the OTS/CafeOBJ method (from the author’s personal viewpoint) are summarized as follows:

- **Equation based specification and verification.** In the OTS/CafeOBJ method, the specifications of OTS models are written in terms of equations and the verification of OTSs is based on equational reasoning. As to the specifications, equations are the most basic logical formulas, which make the specifications easy to understand. As to the verification, equational reasoning is the most fundamental way of reasoning, which can moderate the difficulties of proofs that might otherwise too hard to understand. Besides, no sophisticated knowledge is required to use the OTS/CafeOBJ method.

- **Potentiality for Systematization.** Structural induction is the mainly used technique for reasoning in the OTS/CafeOBJ method, which can be used very systematically. Although case splitting and lemma discovery, which need human verifier’s intelligence, are usually needed to conduct structural induction, they can be conducted systematically because equations are used to characterize (sub-)cases and auxiliary lemmas can be derived from those equations characterizing a (sub-)case. The systematization potentiality of structural induction of the OTS/CafeOBJ method also motivates us to further consider the possibility of combining it with model checking techniques, where the combination as a whole can be made systematic to some extent.

1.4 Thesis Organization

- Chapter 2 provides preliminary knowledge on Domain Methodology, CafeOBJ algebraic specifications, the OTS/CafeOBJ method, and application example in e-Government system analysis. It focuses on introducing how the structural induction of the OTS/CafeOBJ method works for software design/specification verification of invariant properties by using the CafeOBJ system as an interactive theorem prover.

- Chapter 3 describes a general e-Government Model; presents the domain understanding of it; and gives the CafeOBJ specification of it. The following transparency property discussion is based on this model and the refinement of the cases studies can be proved as well.

- Chapter 4 a literature overview around transparency is presented, the narrative definition of transparency is refined as well. Based on the narrative description of transparency, we give the formal understanding and discussion of transparency based on the general model, which be introduced in Chapter 3.
- Chapter 5 first formalizes three cases in real e-Government applications. And then discuss the simulation proof of the specification for social welfare case with general model, to show the transparency discussion on general model can apply on cases specification as well.

- Chapter 6 proposes a method to improve transparency property based on the previous formal discussion, and employs the formal method to show the improvement as well.

- summarizes the thesis, introduces some related work for highlighting our contribution and discusses the future work.
Chapter 2

Preliminaries

In this chapter we provide preliminary knowledge on following points: Domain Engineering, and how do we understand public administration domain by the domain method; CafeOBJ algebraic specifications, the OTS/CafeOBJ method, we focus on introducing how to use interactive structural induction of the OTS/CafeOBJ method to do software design/specification verification of invariant properties; and present an application case on how do we apply OTS/CafeOBJ method in analyzing a real e-government application case. These contents serve as the basis for the work to be introduced later of this thesis.

2.1 Narrative Description of the Domain

2.1.1 Introduction of Public Administration Domain

By public government we shall following Charles de Secondat, baron de Montesquieu (1689–1755)\(^1\), understand a composition of three powers: the law making (legislative), the law enforcing and the law interpreting parts of public government. The three branches of government. Typically we refer to law making government as the national parliament and local (province and city) councils, law enforcing government as the executive (the administration), and law interpreting government as the judiciary system (including lawyers etc.).

We illustrate public administration as a cycle of activities from: citizens empowering their parliament and enhance government; to tackle societal problems by lawmaking; to enforce the administration and these laws by law enforcing; to improve Public Government by Law Interpreting. During the procedure of citizen interacting with the public government branches, different kinds of public documents are generated. The documents may include authorization document, certification document or public operation document and so on.

So we view public administration domain from lawmaking government and its documents via law enforcing government and its documents to law interpreting government and its documents. Different kinds of operation can be applied on documents, their creation, editing, copying, distribution, searches, calculations and shredding and tracing, authorization, and licensing of Document handling. A model of Essential Public Government is as Figure 2.1.

\(^1\)De l’esprit des lois (The Spirit of the Laws), published 1748
We thus include the topic of public administration for a number of view points. That is so as to get as multi-facets and full-scale a view as possibly. The formal description of Chap.3 that capture all of Chap.2, but view it as a system way. In the next two subsection, we are going to informally describe and analyze the domain of public administration. These subsections understand the domain from different viewpoints. First, we examine the phenomena and concept of the domain, entity, function, event, behavior. After present basic phenomena of the domain in subsection 2.2, in section 2.3, are the stakeholder in domain and business process. We repeat important aspects of the domain by our understanding from the point of view of facet.

2.1.2 Entities, Functions, Events and Behaviors

One aspect of design a domain is to give a rough sketch of the Entities, Functions, Events and Behaviors in this domain [17].

By an entity we shall loosely understand something fixed, immobile or static. Although that thing may move, after it has moved it is essentially the same thing, an entity.

By a function we shall loosely understand something, a mathematical quantity (that no one has ever seen), which when applied to something (else), called an argument of the function, yields something (yet else), called a result of the function for that argument. If the function is applied to something which is not a proper argument of the function, then the totally undefined result, called chaos, is yielded.

By an event we shall loosely understand the occurrence of something that may either trigger an action, or is triggered by an action, or alter the course of a behavior, or a combination of these.

By a behavior we shall loosely understand a sequence of actions and events.
Entities

We initially identify the following entities:

- Parliamentary and Parliamentary Committee as Law maker;
- Central Administration Ministries and Local government as Law enforcer;
- Lower Courts and Higher Courts as Law interpreter;
- Citizens who interact with all above three branches of government;
- Document, which we could, instead of the above five classes of entities, consist the first four as behaviors (only) and the last as one entity.

We could, perhaps, identify further entities, but the above are sufficient to make our point.

We divide the entity into two kinds as: atomic entity and composite entity

- By an atomic entity we shall understand an entity which cannot be understood as composed from other entities.
- By a composite entity e we shall understand an entity which can best be understood as composed from other entities, called the sub-entities, e1, e2, ..., en, of entity e.

So here is the description of entities in Public Administration Domain.

Parliamentary Subcommittee: is a composite entity. It is a sub set of the members of parliament. So the mereology of a parliamentary subcommittee is that it has a set of sub-entities: members. The attributes of a parliamentary subcommittee are the name of the parliamentary subcommittee and the member of parliamentary subcommittee.

Parliament: is a composite entity. It is a set of members of parliamentary and a set of parliamentary subcommittee. So the mereology of a parliamentary is that it has two parts are also the sub-entities: the members and the parliamentary subcommittee. The attributes of a parliamentary are the name of the parliamentary, the number of members and the number of parliamentary subcommittee.

Central Administration Ministry: is a composite entity. It is composited by law office and rule and regulation office. So the mereology of a ministry is that it has two parts are also the sub-entities: the law office and the rule and regulation office. The attributes of a ministry are the name of ministry.

Local Government: is a atomic entity. The attributes are: Name of the local government.

Lower Court: is a atomic entity. The attributes are: Name of the lower court

Higher Court: is a atomic entity. The attributes are: Name of the higher court

Citizen: is a atomic entity. The attributes are: Name of the Citizen

Documents: is a atomic entity. The attributes are: Name of the document, Document version number, Document classes(generic and special), History trace of operations(operation,location,time,actor), Reference to relevant license, Name of the field, Value of the field, Text of the field, Name of the sub-field.

Functions

We initially identify the following functions:

- Document Operation: Documents are created, are edited and can be read, copied, moved, can be the basis for searches and calculations, and can be shredded - by actors.
- Document Authorisation: Actors have varying degrees of creation, editing, reading, copying, distribution and shredding authority. And it is be given between actors.
We try to lift all functions as the above two generic functions and other function can be viewed as overload of these functions:

\( \text{Operation\_Doc} : \text{Agent} \times \text{Action} \times \text{Document} \rightarrow \text{Document} \)

\( \text{Authorisation\_Doc} : \text{Agent} \times \text{Action} \times \text{Agent} \times \text{Document} \rightarrow \text{Document} \)

The arguments of above functions are as following:

\begin{itemize}
  \item Agent: Parliamentary Subcommittee, Parliament; Central Admini. Ministries, Local Government; Citizen; Lower Courts, Higher Courts
  \item Document: Problem Formulate, Law Proposal, Discussion (input, output), Law, Rule and Regulation, Handling Procedure, Form(template, fillin), Response, Complaint, etc.
\end{itemize}

The overload of above functions are as following:

\begin{itemize}
  \item formulate: Parliamentary subcommittee and ministry get the information document and yields problem formulate document.
  \item discuss: Parliamentary subcommittee discuss the problem formulate document and yields discuss input document and law proposal
  \item submit: Parliamentary subcommittee submit the yielded documents to parliament and get answer if accept the document or not
  \item debate: Parliament debate the discuss input document and law proposal document and yield law and discuss output document
  \item submit: Parliament submit the yielded documents to ministry and get answer.
  \item enact\_law: Ministry generate the handling procedure by the corresponding law document
  \item submit: Ministry submit the handling procedure to the local government.
  \item operationalize: Local government deal with the handling procedure and create rule and regulation document.
  \item get\_form: Citizens get information from local government generate template form set
  \item act\_wrt\_law: Citizens fill in the template form and yield form set.
  \item submit: Citizens submit the form set to local government and get answer if accept or not
  \item handle: Local government handle the form set and create Local government decision document submit: Local government submit the Local government decision to citizen and citizen decide if accept or not
  \item complaint: Citizens give the local government and judicial system complaint information and get the answer if the complaint be accept or not.
  \item manageable: Judicial system and local government discuss the information and generate Judicial system document
  \item submit: and the judicial system submit the decision to citizen and local government and get corresponding response.
\end{itemize}

We could, perhaps identify further functions, but the above are sufficient to make our point.

**Events**

We initially identify the following events as examples: Law could not be passed; Citizen broken the law; Judiciary System comes down with dew; Citizen applied public service.

Law could not be passed: A parliament committee discusses a specific societal problem and their deliberations are “sent” as a law proposal to parliament which debates the issue
Citizen broken the law: The citizen is contacted by the local government and asked to apply report something on some issue and the citizen replies does not obey the rule and regulation document.

Judiciary System comes down with dew: The citizen does not accept the decision which the local government made and complains to the courts; the “due process of law” take place; and eventually the judiciary system hands down a decision which in favor of some agents or both.

Citizen applied public service: The citizen apply some public service by filling corresponding application form, based on the rule and regulation which published by public government.

We could, perhaps identify further events, but the above are sufficient to make our point.

Behaviors

Take law making procedure as example of behavior. First Parliamentary subcommittee formulate Law Proposal and submit to Parliamentary. Parliamentary discuss the Proposal, and either reject or accept it. Parliamentary subcommittee will reformulate the document if it be rejected.

formulate
discuss
submit $\rightarrow$ reject
re-formulate
formulate
discuss
submit $\rightarrow$ accept
debate
submit $\rightarrow$ OK
enact law

Take law act as example. When citizen create document to fulfil law. Public administration section handle it and give decision back. Citizen does not satisfy with the decision and complaint to local court. Justice system give their decision back to both citizen and public administration section. So the behavior of this process is as follow:

act_wrt_law
submit
handle
decision
complaint
manageable_the_law

We could, perhaps identify further behaviors, but the above are sufficient to make our point.

2.1.3 Other Domain Aspects

The previous section 2.1.2 describes the public administration domain from the point of view of entities, functions, events and behaviors. Now we describe the same domain
from the point of designations, explicit definition, refutable assertion. This is for further understand the domain as well as check if we miss any important points in previous section.

**Designations**

By a *design*, we pragmatically mean a text which ”points” to a phenomenon: something which is manifest: can be sensed and measured.

By a *designation* description(for short: designation) we syntactically mean a textual triple: a designation term, a designation recognition rule, and a designation identification.

a. Designated term: Template Form
   Recognition rule: Forms have not been filled in
b. Designated term: Filled_in_Form
   Recognition rule: Forms have been filled in by the corresponding rules and regulations
c. Designated term: Submission Document
   Recognition rule: Document be submitted to an agent and get accept or reject as respond
d. Designated term: Accept Document
   Recognition rule: Document is accepted by the agent and go on to next handling stage
e. Designated term: Reject Document
   Recognition rule: Document is rejected by the agent and can not go to next handling stage.

**Explicit Definitions**

Designations represent one form of definition. In any description all terms that are special to the universe of discourse being described must be defined either through designations or by explicit definition.

By an explicit definition is understand a definition, i.e, a text which introduces concept, usually names of classes of phenomenon, a class of concepts.

A document is either a template or a form or just some text.

**Refutable Assertions**

By a *refutable assertion* we mean a claim that may be shown to be wrong.

Law is the document which has been discussed and accepted by parliament.

Rule and regulation is the document which has been generated by local government from corresponding law.

**Domain Stake-holders**

By a *domain stakeholder* we shall understand a person, or a group of persons, united somehow in their common interest in, or dependency on the domain; or an institution, an enterprise or a group of such, (again) characterised (and, again, loosely) by their common interest in or dependency on the domain.

Politician, who proposes the social issue, should be deal with.

Ministry, prepare drafts of law document to be put before parliament; formulate and ministers sign rules and regulations on how civil servants are to administrate these laws documents.
Parliament, creation of these laws takes place in parliament; 
Parliamentary committees discuss these law documents and may recommend changes 
or adoption by the parliament. 
Public administration divisions & Administrators, may further amend the rules and 
regulations in these documents with respect to particular interpretation. 
Citizens, who abide (obey or not obey) the laws. 
Officials & Polices, who handle corresponding document and maintain the enforcement 
of the law.

**Business Procedures**

By a business process we understand the procedurally describable aspects, of one or more 
of the ways in which a business, an enterprize, a factory, etc., conducts its yearly, quarterly, 
monthly, weekly and daily processes, that is, regularly occurring chores. 
Public Administration Business Processes: The main business process behaviors of 
a public administration system are the following: The parliamentary Committee, Par- 
lliament, Central Admin. Ministries, Local Government, Citizen, Lower Courts, Higher 
Courts. We describe each of these behaviors separately:

Citizens through the process of devate provoke their parliament to discuss societal 
problems.  
A parliament committee discusses a specific societal problem. 
Their deliberations are sent as a law proposal to parliament which debates the issue 
and passes some law.  
The law is passed on to an appropriate ministry and that ministry formulates basic 
rules and regulations for how local governments shall administrate uses of the law. 
The local governments make provisions for handling the law locally. 
The citizen is either contacted by the local government And asked to report on some 
issue And the citizen replies. 
or the citizen contacts the local government in order to apply for something and the 
local government replies.  
The citizen is either accepts the decision of local government. 
or the citizen does not accept the decision and complains to the courts. 
The due process of law takes place. 
Eventually the judiciary system hands down a decision Either in favor of the citizen 
Or in favor of the government Or both. 
Citizens may direct a problem petition to parliament in the form of a document signed 
by many citizens. 
Parliament decides to do something about the problem. A response document is pro- 
duced. 
A designated parliament committee requests an appropriate ministry to prepare some 
background document. 
The parliament committee passes discussion and a law proposal documents to parlia- 
ment. 
Parliament requests further background documents from the central administration, 
and receives these. 
Parliament debates the law proposal and passes a law, which, as a document is sent to
the appropriate ministry for further handling—and otherwise published in the law gazette.

The ministry and its departments, i.e., the central administration, formulate procedures for the enforcement of the law and sends these, as documents to local administrations.

A citizen applies for some permission, that, is an application document is sent to a local administration or a citizen breaks the law.

The local administration sends a receipt document, possibly forwards further documents to be filled in, and gives a conditional date by which a decision can be expected.

The citizen sends in the possibly further requested documents.

The local administration communicates various documents related to the case to/from other public government offices.

And finally the citizen receives a response document. The citizen

- either sends an acceptance document to the local administration; b. or rejects it, informing the local administration of this, and directs a complaint at the law courts.

The first instance law court deliberates.

A decision is sent to the citizen and the local admin, Neither the local administration or the citizen both accepts the decision and further actions are curtailed, or at least one of them appeals the decision.

Lower court decision documents are passed on to a higher court.

And steps “instance law” and “decision” are repeated till a final decision is passed. From the above rough sketches of behaviors the domain engineer then goes on to describe types of messages between behaviors, types of entities specific to the behaviors, and the functions that apply to or yield those entities.

Facets

By a domain facet we understand one amongst a finite set of generic ways of analyzing a domain: a view of the domain, such that the different facets cover conceptually different views, and such that these views together cover the domain.

In this section we identify a number of domain facets and we survey principles and techniques for modeling, relative to identified domain stakeholder classes, each of the identified facets. So far we have been able to identify the following facets:

- intrinsics,
- support technology,
- management and organization,
- rules and regulations including,
- script facets, and
- human behavior.

We enlarge upon the above enumeration using the following brief characterizations:

**Domain intrinsics**: that which is common to all facets.

**Domain support technologies**: That in terms of which several other facets (intrinsics, business processes, management and organization, and rules and regulations) are imple-
mented.

**Domain management and organization**: That which primarily determines and constrains communication between enterprise and stakeholder.

**Domain rules, regulations and scripts**: That which guides the work of enterprise stakeholder, their interaction, and the interaction with non-enterprise stakeholders.

**Domain human behavior**: The way in which domain stakeholder dispatch their actions and interactions wrt. the enterprise: dutifully, forgetfully, sloppily, yes, even criminally.

To help us identify parts of the above facets we suggest that rough sketch descriptions first be made of what we shall call the domain business process facilitators.

**Intrinsics**

By domain *intrinsics* we shall understand those phenomena and concepts of a domain which are basic to any of the other facets (listed earlier and treated, in some detail, below), with such domain intrinsics initially covering at least one specific, hence named, stakeholder view.

Public Administration Intrinsics: We abstract the Public Administration System as different kind of stakeholder dealing with different instances of document.

So the intrinsic entity is document. And the intrinsic functions are the functions which map to different kinds of document handling:

a, create (formulate): the create function, when applied to an problem issue, yields a document which writes that provides information about public administration.

b, distribute (submit): the distribute function, when applied to a document, yield many copies of the exact same document.

c, complete (handle): the complete function, when applied a document, yield another kind of document. The intrinsic events can be considered as a document is distributed which map to the document is submitted.

One of the intrinsic behavior is the document be created and distribute as the problem formulate document be formulated and submit to discuss.

**Supporting Technologies**

By domain *support technology* we shall understand ways and means of implementing certain observed phenomena.

In the old day the documents are mainly papery and be distributed by manual. By the developing of Information technology, documents become paper-less and digital form, so that can be distributed by internet.

The monitors to watch who break the law are mainly human (such as police) and now it can be recorded by video monitor.

**Management and Organization**

By domain management we shall understand such people who determine, formulate and thus set standards concerning strategic, tactical and operational decisions.

Public Administration Management and Organization:

Making Laws: 1. Citizens through the process of debate provoke their parliament to discuss societal problem. 2. A parliament committee discusses a specific societal problem.
3. Their deliberations are “sent” as a law proposal to parliament which debates the issue and passes some law.

   Enforcement Laws: 4. The law is passed on to an appropriate ministry (of the central government) and that ministry formulates basic rules and regulations for how local governments shall administrate uses of the law. 5. The local governments makes provisions for handling the law locally. 6. The citizen is either: contacted by the local government and asked to report on some issue, and the citizen replies. 7. or the citizen contacts the local government in order to apply for something and the local government replies.

   Interpreting Laws: 8. The citizen is either accepts the decision of local government 9. or the citizen does not accept the decision, and complains to the courts. 10. The “due process of law” takes place. 11. Eventually the judiciary system hands down a decision either in favor of the citizen, or in favor of the government, or both.

Rules and Regulations

By a domain rule we shall understand some text which prescribes how people or equipment are expected to behave when dispatching their duty, respectively when performing their function.

By a domain regulation we shall understand some text which prescribes what remedial actions are to be taken when it is decided that a rule has not been followed according to its intention.

Making Laws: Rule: In China the Law can be passed in parliament is subject to the following rule:
The law can only be passed by two of third members vote for the law proposal. Regulation: If it is discovered that the above rule is not obeyed, then there is some regulation which prescribes administrative or legal management and/ or staff action, as well as some correction to the Law Making.

Scripts

Public Administration Scripts (it may consider as a script language as license, which describe different actors have different kinds of right to handle different types of documents). And an actor can either issue, receive, perform or just halt:

Issue Script:
routine: Actor give license to other actors to handle documents.

Receive Script:
routine: Actor get license to deal with corresponding document.

Perform Script:
routine: the actor should first select a license to perform and than perform the corresponding operation in this license.

Human Behavior

Public Administration Staff Behavior:

Let us assume a police, when enforce a law. We would characterize such a police as being diligent, etc., if that person carefully follows the rule and regulation document, and check the citizens strictly if they following the document items or not. We would characterize a police as being sloppy if that person occasionally forgets the duty above.
2.1.4 E-Government with Public Administration Domain

The figure 2.2 illustrates the action sequences in public administration domain. By the general understanding of e-Government, e-government provides public service by information and communication technologies; we understand the most important actions which be involved in the e-government party, are the branches law enforcer interacts with citizen. Some government system provides citizen interact with law interpreter. And partially law making procedure actions are provided by government system.

Therefore, citizen, as an important stakeholder in public administration domain, they are mainly involved in law enforcing processing (administrative processing) of current government systems. How about the desired property embedded in this part, and by the system design, how shall we consider the property in other two branches, are what we want to achieve by the rest of this research.

2.2 The OTS/CafeOBJ Methods

The OTS/CafeOBJ method [40, 48, 41] is a modeling, specification and verification method. In the OTS/CafeOBJ method, a system to be verified is first modeled as an observational transition system (OTS), a transition system that can be straightforwardly written in terms of equations; The OTS is then written in CafeOBJ as a behavioral specification; Desired safety and liveness properties of the OTS can be verified by means

---

2Some basic data types used in the OTS, such as `Nat` and `Int` in the bank account system, are described as general algebraic specifications, which are imported in this behavioral specification. Note that actually `Nat` and `Int` are predefined data types of CafeOBJ, thus can be directly imported and used.
of writing and executing proofs (called proof scores) in CafeOBJ by using the CafeOBJ system as an interactive theorem prover.

An OTS/CafeOBJ specification consists of a list of modules such that: one module specifies an OTS (the behavioral specification), called OTS module, which is the core module of the OTS/CafeOBJ specification; some specify data types used in the OTS module, called data type modules; and one specifies the properties to be proved, called property module.

Since the only properties considered in the thesis are invariant properties, which are the most basic and important properties among various kinds of properties. So all the aspects of the OTS/CafeOBJ method that are related to liveness properties are omitted (for those contents about liveness properties, please refer to [48]), and thus the property module in an OTS/CafeOBJ specification is called invariant module.

2.2.1 Observational Transition Systems

The notion of OTSs [40] is used as the formal model in this method. We assume that there exists a universal state space called $\mathcal{Y}$, and that data types used, including the equivalence relation (denoted by $=$) for each data type, have been defined in advance. An OTS $S$ consists of $\langle \mathcal{O}, \mathcal{I}, \mathcal{T} \rangle$:

- $\mathcal{O}$: A finite set of observers. Each $o \in \mathcal{O}$ is a function $o : \mathcal{Y} \to D$, where $D$ is a data type and may differ from observer to observer. The equivalence relation between two states $\upsilon_1, \upsilon_2 \in \mathcal{Y}$ wrt $S$ (denoted by $\upsilon_1 =_S \upsilon_2$) is defined as $\forall o \in \mathcal{O}, o(\upsilon_1) = o(\upsilon_2)$.

- $\mathcal{I}$: The set of initial states such that $\mathcal{I} \subseteq \mathcal{T}$.

- $\mathcal{T}$: A finite set of conditional transitions. Each $t \in \mathcal{T}$ is a function $t : \mathcal{Y} \to \mathcal{Y}$, provided that $t(\upsilon_1) =_S t(\upsilon_2)$ for each $[\upsilon] \in \mathcal{Y}/ =_S$ and each $\upsilon_1, \upsilon_2 \in [\upsilon]$. $t(\upsilon)$ is called the successor state of $\upsilon \in \mathcal{Y}$ wrt $t$. The condition $c_t$ of $t$ is called the effective condition. For each $\upsilon \in \mathcal{Y}$ such that $\neg c_t(\upsilon)$, $\upsilon =_S t(\upsilon)$.

Observers and transitions may be parameterized. Generally, observers and transitions are denoted by $o_{x_1, \ldots, x_m}$ and $t_{y_1, \ldots, y_n}$, provided that $m, n \geq 0$ and there exist corresponding data types $D_k \ (k = x_1, \ldots, x_m, y_1, \ldots, y_n)$.

The formal analysis in the OTS/CafeOBJ method is essentially analysis of the reachable states wrt $S$, which are inductively defined as follows: (1) each $\upsilon_0 \in \mathcal{I}$ is reachable, and (2) for each $t \in \mathcal{T}$, $t(\upsilon)$ is reachable if $\upsilon \in \mathcal{Y}$ is reachable. The properties that we considered in this experiment are all invariant (properties). An invariant wrt $S$ is a state predicate $p : \mathcal{Y} \to \text{Bool}$, which holds in all reachable states wrt $S$.

2.2.2 Description of OTSs in CafeOBJ

In the OTS/CafeOBJ method, an OTS is described in CafeOBJ [9, 62]. CafeOBJ is an algebraic specification language and system mainly based on order-sorted algebras and hidden algebras. Data types can be specified in terms of order-sorted algebras, and state machines such as OTSs can be specified in terms of hidden algebras. A CafeOBJ visible sort denotes an abstract data type, and a hidden sort denotes the state space of an abstract state machine. Both the abstract data types and the state machine are described as modules, which are the basic building blocks of CafeOBJ specifications. There are two
kinds of operators in hidden sorts: action and observation operators. An action operator can change a state of an abstract state machine, and only observation operators can be used to observe the inside of an abstract state machine. Declarations of observation and action operators start with \texttt{bop}, and those of other operators with \texttt{op}. Operators are defined in equations. Declarations of equations start with \texttt{eq}, and those of conditional equations with \texttt{ceq}. The CafeOBJ system rewrites a given term by regarding equations as left-to-right rewrite rules.

The universal state space \( \Upsilon \) is denoted by a hidden sort, say \( H \). An observer \( o_{x_1, \ldots, x_m} \) is denoted by a CafeOBJ observation operator and declared as \texttt{bop} \( o : H \rightarrow V \), where \( V_{x_1}, \ldots, V_{x_m} \) and \( V \) are visible sorts.

Any state in \( \Upsilon \) (namely any initial state) is denoted by a constant, say \( \textit{init} \), which is declared as \texttt{op} \( \textit{init} : \rightarrow H \). The equation expressing the initial value of \( o_{x_1, \ldots, x_m} \) is as follows:

\[
\text{eq } o(\textit{init}, X_{x_1}, \ldots, X_{x_m}) = f(X_{x_1}, \ldots, X_{x_m}).
\]

\( X_k \) is a CafeOBJ variable of \( V_k \), where \( k = x_1, \ldots, x_m \), and \( f(X_{x_1}, \ldots, X_{x_m}) \) is a CafeOBJ term denoting the initial value of \( o_{x_1, \ldots, x_m} \).

A transition \( t_{y_1, \ldots, y_n} \in T \) is denoted by a CafeOBJ action operator and declared as \texttt{bop} \( t : H \rightarrow V_{y_1}, \ldots, V_{y_n} \), where \( V_{y_1}, \ldots, V_{y_n} \) are visible sorts. \( t_{y_1, \ldots, y_n} \) may change the value returned by \( o_{x_1, \ldots, x_m} \) if it is applied in a state \( v \) such that \( c_{t_{y_1, \ldots, y_n}}(v) \), which can be written generally as follows:

\[
\text{ceq } o(t(S, X_{y_1}, \ldots, X_{y_n}), X_{x_1}, \ldots, X_{x_m}) =
\begin{cases} 
  e(t(S, X_{y_1}, \ldots, X_{y_n}, X_{x_1}, \ldots, X_{x_m}) & \text{if } e(t(S, X_{y_1}, \ldots, X_{y_n})) \\
  c\text{-}t(S, X_{y_1}, \ldots, X_{y_n}) & \text{if } c\text{-}t(S, X_{y_1}, \ldots, X_{y_n}) \text{ changes nothing if it is applied in a state } v \text{ such that } \neg c\text{-}t(S, X_{y_1}, \ldots, X_{y_n})(v), \text{ which can be generally written as:}
\end{cases}
\]

\[
\text{ceq } t(S, X_{y_1}, \ldots, X_{y_n}) = S \text{ if not } c\text{-}t(S, X_{y_1}, \ldots, X_{y_n}).
\]

\[2.2.3 \quad \text{Analysis of OTSs}\]

\textbf{Falsification with Search Command}

CafeOBJ system provides a search command that can be used to falsify (find logical errors of) invariants of OTSs specified in CafeOBJ specification language. General idea of using this command for falsification is trying to explore the reachable state space of an OTS from its initial state to a state in which an invariant does not hold.

To use the CafeOBJ search command, we need first to give an explicit state structure for an OTS \( S \), and then obeying this state structure, we give, for each action (representing a transition of \( S \)) defined in equations, additional \textit{state transition} expressions. Assume that the state structure of an OTS is decided to be \( \langle v \rangle \), then an additional
state transition expression for the action expression described in Subsection 3.2 (the second equation) can be generally written in a module TRANS (which imports the module where $S$ is defined) as follows:

$$\text{ctrans} < S > \Rightarrow < t(S, i_{y_1}, \ldots, i_{y_n}) > \text{ if } c-t(S, i_{y_1}, \ldots, i_{y_n}).$$

$\text{ctrans}$ is a keyword to declare state transition expressions. $i_{y_1}, \ldots, i_{y_n}$ are CafeOBJ constants of corresponding variables $X_{y_1}, \ldots, X_{y_n}$. These constants are determined and chosen by human analyzers with the purpose to make the state space to be explored smaller. Therefore according to the number of constants selected for each variable, several such state transition expressions may exist.

A CafeOBJ search command is in the following form:

$$\text{red} < \text{init} > = (m, n)\Rightarrow* < S > \text{ suchThat } \text{pred}(S).$$

The command returns true if a state $S$ satisfying the condition $\text{pred}(S)$ is reached, via more than 0 time transitions, from the initial state $\text{init}$. Otherwise, it returns false. $n$ is the upper bound of the number of applied transitions, and $m$ is the upper bound of the number of such state $S$. For falsification purpose, we could put the negation of the invariant to be analyzed in the position of $\text{pred}(S)$.

**Verification with Proof Scores**

To verify (prove correctness of) invariants of an OTS $S$, we generally need to do induction on the reachable state space of $S$. We describe how to prove a predicate $p_1$ is invariant to $S$ using induction by writing proof scores [40] in CafeOBJ. The proof that $p_1$ is invariant to $S$ often needs other predicates. We suppose that $p_2, \ldots, p_n$ are such predicates. We then prove $p_1 \land \ldots \land p_n$ is an invariant to $S$. Let $X_{i_1}, \ldots, X_{i_m}$ of types $D_{i_1}, \ldots, D_{i_m}$ be all free variables in $p_i$ ($i = 1, \ldots, n$) except for $v$ whose type is $\text{T}$.

We first declare the operators denoting $p_1, \ldots, p_n$. Their defining equations in a module INV (which imports the module where $S$ is defined) are written as follows:

$$\text{op inv}_i : H V_{i_1} \ldots V_{i_m} \rightarrow \text{Bool}$$
$$\text{eq inv}_i(S, X_{i_1}, \ldots, X_{i_m}) = p_i(S, X_{i_1}, \ldots, X_{i_m}).$$

where $i = 1, \ldots, n$. $V_k$ ($k = i_1, \ldots, i_m$) is a visible sort denoting $D_k$, and $X_k$ is a CafeOBJ variable whose sort is $V_k$. $p_i(S, X_{i_1}, \ldots, X_{i_m})$ is a CafeOBJ term denoting $p_i$. In INV, we also declare a constant $x_k$ denoting an arbitrary value of $V_k$. We then declare the operators denoting basic formulas to show in the inductive cases and their defining equations in a module ISTEP (which imports INV) as follows:

$$\text{op istep}_i : V_{i_1} \ldots V_{i_m} \rightarrow \text{Bool}$$
$$\text{eq istep}_i(X_{i_1}, \ldots, X_{i_m}) = \text{inv}_i(s, X_{i_1}, \ldots, X_{i_m}) \implies \text{inv}_i(s, X_{i_1}, \ldots, X_{i_m}).$$

3Generally the state space of an OTS is infinite. Exploring an infinite state space will cause the search command to keep running until no enough memory. This is called the state explosion problem for general state explosion techniques.
where \( i = 1, \ldots, n \). \( s \) and \( s' \) are constants of sort \( H \), denoting an arbitrary state and a successor state of \( s \).

The proof of each inductive case often requires case analysis. Let us consider the inductive case where it is shown that \( t_{y_1,\ldots,y_n} \) preserves \( p_i \). Suppose that the state space is split into \( l \) sub-spaces for the proof of the inductive case, and that each sub-space is characterized by a predicate \( \text{case}_k \) \((k = 1, \ldots, l)\) such that \( (\text{case}_1 \lor \ldots \lor \text{case}_l) \leftrightarrow \text{true} \).

Also suppose that visible sorts \( V_{y_1}, \ldots, V_{y_n} \) correspond to data types \( D_{y_1}, \ldots, D_{y_n} \) of the parameters of \( t_{y_1,\ldots,y_n} \). The proof for case \( \text{case}_k \) is shown as follows:

```
open ISTEP
  -- arbitrary objects denoted by constants e
op e_{y_1} :-> V_{y_1}. \ldots \ldots \op e_{y_n} :-> V_{y_n}.
  -- assumptions
  Declarations of equations denoting \( \text{case}_k \).
  -- successor state
eq s' = t(s,e_{y_1},\ldots,e_{y_n}).
  -- check if the predicate is true
red SIH_i implies istep_i(x_{i_1},\ldots,x_{i_m}).
close
```

where \( i = 1, \ldots, n \). A comment starts with \(--\) and terminates at the end of the line. \( SIH_i \) is a CafeOBJ term denoting what strengthens the inductive hypothesis \( \text{inv}_i(s,x_{i_1},\ldots,x_{i_m}) \) and can be the concatenation of different predicates ranging from \( \text{inv}_1(\ldots) \) to \( \text{inv}_n(\ldots) \). The CafeOBJ command \( \text{red} \) is used to reduce a term denoting a proposition to its truth value. \( \text{open} \) creates a temporary module that imports a module given as an argument, and \( \text{close} \) destroys the temporary module. Parts enclosed with \( \text{open} \) and \( \text{close} \) are basic units of proof scores, which are called proof passages.

### 2.3 Application Example of OTS/CafeOBJ Method in Government System Analysis

#### 2.3.1 Application Background

One major responsibility of e-Government initiatives is to provide citizens with public services electronically through information and communication technologies. It is extremely important and necessary for the system implementation for e-Government to be reliable. Unreliable system implementation could, on the one hand, be insufficient to fulfill the basic system requirements, and more seriously on the other hand, break the trust of citizens on governments.

The objective of this example is to make a same advocation as the one by Jim Davies [?], \textit{et al.}, but we back up this advocation with an experiment on using a specific formal method – the OTS/CafeOBJ method [40] – to an e-Government messaging framework [21, 20, 22]. The framework is proposed for providing citizens with seamless public services\(^4\), and its key idea is to support asynchronous message exchange among reg-

\(^4\)Seamless public services allow citizens to specify a need and obtain a service to fulfill this need
istered members (government agencies) through dynamically created channels. In [22], a formal specification of the framework using the RAISE specification language (RSL [72]) has been provided but without formal analysis.

In our experiment, the framework is first modeled as an OTS (Observational Transition System) [40], a kind of state transition system that could be straightforwardly written in terms of equations. The OTS is then specified with CafeOBJ algebraic specification language [9, 62]. The resulted specification of the framework is consequentially analyzed with both falsification and verification mechanisms supported by CafeOBJ system. We use the term falsification to denote finding logic errors, and verification to denote proving correctness. With falsification, we found two previously not well-clarified problems of the framework. We analyzed potential harm that could be caused by the problems and suggested possible ways of revisions to the framework. Furthermore with verification, we proved that the revised framework satisfies certain desired properties. Therefore based on this experiment, we could make a more particular argument that e-Government initiatives would benefit from formal falsification and verification (analysis) activities.

### 2.3.2 A Messaging Framework

We outline the main part of the messaging framework and refer the readers to [21, 20, 22] for more detailed information. Key idea of the framework is to support asynchronous message exchange among registered members (government agencies) through dynamically created channels. The framework comprises three components: (1) core of the framework that fulfills the basic message exchange functionalities, named as MSG-CORE, (2) extensions that enable additional functionalities to the core, named as MSG-EXTD. Example extensions include logging of messages passing through a channel, and encryption/decryption of messages etc, and (3) a development methodology for building other new extensions, named as MSG-METH. We only focus on MSG-CORE as [22] did.

An illustration of MSG-CORE is shown in Figure 1. We informally outline the functionalities of MSG-CORE as follows:

- **register** – given an identifier \( i \), a government agency registers a member with \( i \) as the member’s identifier (for simplicity, we will call it member \( i \)). Its precondition requires that \( i \) is currently not in use by other members. Each member has a pair of inbox and outbox containing lists of incoming and outgoing messages for this member, which are initially empty.

- **unregister** – given an identifier \( i \), the government agency (that owns member \( i \)) unregisters the member \( i \). Its precondition requires that member \( i \) exists and does not own any channel.

- **create** – given two identifiers \( i \) and \( x \), member \( i \) creates a channel \( x \). Member \( i \) will be the owner of channel \( x \), and is a subscriber to channel \( x \). Each channel maintains a record of identifiers who subscribed to the channel. Its precondition requires that member \( i \) exists and \( x \) is not currently in use by other channels.

- **destroy** – given two identifiers \( i \) and \( x \), member \( i \) destroys channel \( x \). Its precondition requires that channel \( x \) exists, member \( i \) is the owner of channel \( x \), and channel \( x \) does not have other subscribers except \( i \).

without knowing which agency or level of government should be contacted [20]. Such a service is usually delivered by collaboration of multiple government agencies.
• **subscribe** – given two identifiers $i$ and $x$, member $i$ subscribes to channel $x$. $i$ will be added into channel $x$’s subscriber record. Its precondition requires that member $i$ and channel $x$ exist, and $i$ is not already in channel $x$’s subscriber record.

• **unsubscribe** – given two identifiers $i$ and $x$, member $i$ unsubscribes to channel $x$. $i$ will be removed from channel $x$’s subscriber record. Its precondition requires that member $i$ and channel $x$ exist, and $i$ is in channel $x$’s subscriber record.

• **send** – given an identifier $i$ and a message $msg$, member $i$ sends a message $msg$ to member $i$’s outbox. Its precondition requires that member $i$ exists, and $i$ is recorded in $msg$ as the sender. Each message contains the information of sender, channel (through which the message will be delivered to its receivers), and receivers (a set of member identifiers).

• **deliver** – given an identifier $i$, a message in member $i$’s outbox is sent out to the message’s receivers’ inboxes through a channel requested by the message. The information about channel and receivers could be obtained from the message. Its precondition requires that member $i$ exists, its outbox is not empty, the channel exists, member $i$ is a subscriber of the channel, the receivers exist, and the receivers are subscribers of the channel. A sent message will be removed from the corresponding sender’s outbox.

• **receive** – given an identifier $i$, a message in member $i$’s inbox is obtained by $i$ and removed from the inbox. Its precondition requires that member $i$ exists and its inbox is not empty.

**MSG-CORE** maintains a record of existing members and channels (their identifiers). When register (unregister) a member, the member’s identifier is added into (removed from) the record. Similar things happen to the record when create (destroy) a channel. Besides, we do not consider government agencies in our experiment, and just consider a member is registered (unregistered).

As shown in Figure 1, there is a special member **admin**, which owns a channel for each member. **admin** is responsible for administrating the framework, such as registering new members etc. In this work we omit this part for simplicity.

### 2.3.3 Modeling and Specifications

We start our experiment on modeling **MSG-CORE** as an OTS, called $S_{core}$, and then specifying $S_{core}$ in CafeOBJ.

The data types used in $S_{core}$ are:

- **Bool** for truth values.
- **Mname**, **Cname**, and **MsgID** for the names of members and channels, and the identifiers of messages.

- **SetMname** and **SetCname** for the set of member names and the set of channel names, where **empty** denotes an empty set.

- **Message** for messages, where \( \text{mkMsg}(\text{id:MsgID}, m: \text{Mname}, c: \text{Cname}, ms: \text{SetMname}) \) denotes a message (with the identifier \( \text{id} \)) that is to be sent by sender \( m \) through channel \( c \) to receivers \( ms \). Projection functions \( \text{mid}, \text{sender}, \text{chn}, \) and \( \text{receivers} \) are defined to extract corresponding components of a message, respectively. For example, \( \text{mid}(\text{mkMsg}(\text{id}, m, c, ms)) = \text{id} \).

- **MsgList** for a list of messages.

- **Member** for members, where \( \text{mkMem}(m: \text{Mname}, cs1: \text{SetCname}, cs2: \text{SetCname}, ib: \text{MsgList}, ob: \text{MsgList}) \) denotes a member with the name \( m \), who owns a set of channels \( cs1 \), subscribed to a set of channels \( cs2 \), and has the inbox \( ib \) and outbox \( ob \) for messages. Projection functions \( \text{mid}, \text{owns}, \text{subed}, \text{inbox}, \) and \( \text{outbox} \) are defined to extract corresponding components of a member, respectively. \( \text{invalidM} \) denotes an unexisting member.

- **Channel** for channels, where \( \text{mkChn}(c: \text{Cname}, m: \text{Mname}, ms: \text{SetMname}) \) denotes a channel with the name \( c \), which is owned by a member \( m \) and is subscribed by a set of members \( ms \). Projection functions \( \text{cid}, \text{owner}, \) and \( \text{subs} \) are defined to extract corresponding components of a channel, respectively. \( \text{invalidC} \) denotes an unexisting channel.

- **MemChn** denotes a pair of member name set and channel name set, where \( [s1: \text{SetMname}, s2: \text{SetCname}] \) denotes that currently there exist a set \( s1 \) of members (names) and a set \( s2 \) of channels (names). Projection functions \( \text{mSet} \) and \( \text{cSet} \) are defined to extract corresponding components of a pair.

\[
\mathcal{S}_{\text{core}} \text{ is } \langle O_{\text{core}}, I_{\text{core}}, T_{\text{core}} \rangle \text{ such that:}
\]
\[
O_{\text{core}} \triangleq \{ \text{member}_{m: \text{Mname}} : \Upsilon \rightarrow \text{Member},
\text{channel}_{c: \text{Cname}} : \Upsilon \rightarrow \text{Channel},
\text{record} : \Upsilon \rightarrow \text{MemChn} \}
\]
\[
I_{\text{core}} \triangleq \{ v_{\text{init}} \in \Upsilon \mid \text{member}(v_{\text{init}}, m) = \text{invalidM} \wedge
\text{channel}(v_{\text{init}}, c) = \text{invalidC} \wedge
\text{record}(v_{\text{init}}) = [\text{empty}, \text{empty}] \}
\]
\[
T_{\text{core}} \triangleq \{ \text{register}_{m: \text{Mname}} : \Upsilon \rightarrow \Upsilon,
\text{unregister}_{m: \text{Mname}} : \Upsilon \rightarrow \Upsilon,
\text{create}_{m: \text{Mname}, c: \text{Cname}} : \Upsilon \rightarrow \Upsilon,
\text{destroy}_{m: \text{Mname}, c: \text{Cname}} : \Upsilon \rightarrow \Upsilon,
\text{subscribe}_{m: \text{Mname}, c: \text{Cname}} : \Upsilon \rightarrow \Upsilon,
\text{unsubscribe}_{m: \text{Mname}, c: \text{Cname}} : \Upsilon \rightarrow \Upsilon,
\text{send}_{m: \text{Mname}, msg: \text{Message}} : \Upsilon \rightarrow \Upsilon,
\text{receive}_{m: \text{Mname}} : \Upsilon \rightarrow \Upsilon,
\text{deliver}_{m: \text{Mname}} : \Upsilon \rightarrow \Upsilon \}
\]

\( \mathcal{S}_{\text{core}} \) is specified as a module called **MESSAGING** in CafeOBJ specification language. We assume that all the data types used such as **Member**, **Channel**, and **Message** etc. have been
defined as modules in advance, and imported by the module **MESSAGING**. The signature of the module is as follows:

--- any initial state
op **init** : -> **Sys**

--- observation operators
  bop **member** : **Sys** **Mname** -> **Member**
  bop **channel** : **Sys** **Cname** -> **Channel**
  bop **record** : **Sys** -> **MemChn**

--- action operators
  bop **register** : **Sys** **Mname** -> **Sys**
  bop **unregister** : **Sys** **Mname** -> **Sys**
  bop **create** : **Sys** **Mname** **Cname** -> **Sys**
  bop **destroy** : **Sys** **Mname** **Cname** -> **Sys**
  bop **subscribe** : **Sys** **Mname** **Cname** -> **Sys**
  bop **unsubscribe** : **Sys** **Mname** **Cname** -> **Sys**
  bop **send** : **Sys** **Mname** **Message** -> **Sys**
  bop **receive** : **Sys** **Mname** -> **Sys**
  bop **deliver** : **Sys** **Mname** -> **Sys**

**Sys** is the hidden sort denoting the state space **Υ** of **S**\textsuperscript{core}. Constant **init** denotes an arbitrary initial state of **S**\textsuperscript{core}. The three observation operators correspond to the observers, and the nine action operators correspond to the transitions. In this work, we show the CafeOBJ specifications for **init** and actions **unregister** and **unsubscribe** as demonstration examples. The remaining actions could be defined in a similar way.

--- for any initial state **init**
eq **member**(init,P) = invalidM .
eq **channel**(init,X) = invalidC .
eq **record**(init) = [empty,empty] .

The variables **P** and **X** are of sorts **Mname** and **Cname**, denoting an arbitrary member name and an arbitrary channel name, respectively. The meaning of the equations is that in the initial state, there exist no members and channels, and thus the record is a pair of empty sets of members and channels.

--- unregister a member with a given name **P**
op **c-unregister** : **Sys**
Mname -> **Bool**
eq **c-unregister**(S,P)
  = **P** /\in mSet(record(S)) and owns(member(S,P)) = empty.
-- ceq **member**(unregister(S,P),Q)
  = (if (P = Q) then invalidM else **member**(S,Q) fi)
  if **c-unregister**(S,P) .
  ceq **channel**(unregister(S,P),Y)
    = **channel**(S,Y)  if **c-unregister**(S,P) .
  ceq **record**(unregister(S,P))
    = [del(P,mSet(record(S))), cSet(record(S))])
    if **c-unregister**(S,P) .
  ceq **unregister**(S,P) = S if not **c-unregister**(S,P) .

The effective condition of **unregister** demands that: in state **S**, the to-be-unregistered member **P** should exist and it has no owned channels. If the effective condition holds, execution of **unregister** will: change the observed value of **member** to **invalidM** (namely
a member with the name P does not exist); the return value of channel remain unchanged; and remove the name P from the member name set of the record (by the function del). If the effective condition does not hold, nothing changes. Note that the variable P denotes an arbitrary member to be unregistered, and variable Q denotes an arbitrary member that the observer member is “observing” on, and thus we need to compare if they are the same member. Similar comparison is done for channels.

The specification for action unsubscribe is as follows, whose meaning can be explained in a similar way.

\[
\text{-- unsubscribe a channel X by a member P op c-unsubscribe} : \text{Sys}
\]

\[
\text{Mname Cname} \to \text{Bool eq c-unsubscribe}(S,P,X)
\]

\[
= \text{P /in mSet(record(S)) and X /in cSet(record(S))}
\]

\[
\text{and P /in subs(channel(S,X))} .
\]

\[
\text{-- ceq member unsubscribe(S,P,X),Q}
\]

\[
= (\text{if (P = Q)}
\]

\[
\text{then mkMem(Q,owns(member(S,Q))),del(X,subed(member(S,Q))),}
\]

\[
\text{inbox(member(S,Q)),outbox(member(S,Q)))}
\]

\[
\text{else member(S,Q) fi) if c-unsubscribe(S,P,X) .}
\]

\[
\text{ceq channel unsubscribe(S,P,X),Y}
\]

\[
= (\text{if (X = Y)}
\]

\[
\text{then mkChn(Y,owner(channel(S,Y))),}
\]

\[
\text{del(P,subs(channel(S,Y))))}
\]

\[
\text{else channel(S,Y) fi) if c-unsubscribe(S,P,X) .}
\]

\[
\text{ceq record unsubscribe(S,P,X) = record(S) if c-unsubscribe(S,P,X) .}
\]

\[
\text{ceq unsubscribe(S,P,X) = S if not c-unsubscribe(S,P,X) .}
\]

2.3.4 Analysis

We give the informal descriptions of three basic desired properties of $S_{core}$ that we have analyzed.

**Property 1:** All subscribers to a channel are members.

**Property 2:** The owner of a channel is a member.

**Property 3:** The owner of a channel is subscribed to the channel.

The three properties are specified in a module INV (which imports the module MESSAGING defining $S_{core}$) as follows:

\[
\text{op prop1 : Sys Mname Cname } \to \text{Bool eq prop1}(S,P,C)
\]

\[
= C /\text{in cSet(record(S)) and P /in subs(channel(S,C))}
\]

\[
\text{implies P /in mSet(record(S))} .
\]

\[
\text{op prop2 : Sys Cname } \to \text{Bool eq prop2}(S,C)
\]

\[
= C /\text{in cSet(record(S))}
\]

\[
\text{implies owner(channel(S,C)) /in mSet(record(S))} .
\]

\[
\text{op prop3 : Sys Cname } \to \text{Bool eq prop3}(S,C)
\]

\[
= C /\text{in cSet(record(S))}
\]

\[
\text{implies owner(channel(S,C)) /in subs(channel(S,C))} .
\]

28
Falsification of $S_{core}$

Following the introduction in Subsection 3.3.1, we first prepare additional state transition expressions in a module $TRANS$ (which imports the module $INV$, and thus $MESSAGING$ is also imported) for each of the nine actions.

We choose two constants $i$ and $j$ denoting arbitrary member names, and two constants $x$ and $y$ denoting arbitrary channel names. The state transition expressions for $unregister$ and $unsubscribe$ are shown as demonstration examples as follows, in which we give names to the transitions by adding labels such as “[unregister-i]”:

$$\text{ctrans [unregister-i]} : \langle S \rangle \Rightarrow \langle \text{unregister}(S,i) \rangle \text{ if } c-\text{unregister}(S,i).$$

$$\text{ctrans [unregister-j]} : \langle S \rangle \Rightarrow \langle \text{unregister}(S,j) \rangle \text{ if } c-\text{unregister}(S,j).$$

$$\text{-- ctrans [unsubscribe-ix]} : \langle S \rangle \Rightarrow \langle \text{unsubscribe}(S,i,x) \rangle \text{ if } c-\text{unsubscribe}(S,i,x).$$

$$\text{ctrans [unsubscribe-iy]} : \langle S \rangle \Rightarrow \langle \text{unsubscribe}(S,i,y) \rangle \text{ if } c-\text{unsubscribe}(S,i,y).$$

$$\text{ctrans [unsubscribe-jx]} : \langle S \rangle \Rightarrow \langle \text{unsubscribe}(S,j,x) \rangle \text{ if } c-\text{unsubscribe}(S,j,x).$$

$$\text{ctrans [unsubscribe-jy]} : \langle S \rangle \Rightarrow \langle \text{unsubscribe}(S,j,y) \rangle \text{ if } c-\text{unsubscribe}(S,j,y).$$

We then try to falsify the three properties by inputting the following search commands into the CafeOBJ system.

$$\text{red } < \text{init} > = (1,\ast)\Rightarrow \ast < S:Sys > \text{ suchThat (not prop1-ijxy(S))}. \text{ red } < \text{init} > = (1,\ast)\Rightarrow \ast < S:Sys > \text{ suchThat (not prop2-xy(S))}. \text{ red } < \text{init} > = (1,\ast)\Rightarrow \ast < S:Sys > \text{ suchThat (not prop3-xy(S))}. $$

In the above commands, the terms like $\text{prop1-ijxy}(S)$ are instantiated versions of corresponding properties. For example, $\text{prop1-ijxy}(S) = (\text{prop1}(S,i,x) \text{ and } \text{prop1}(S,i,y) \text{ and } \text{prop1}(S,j,x) \text{ and } \text{prop1}(S,j,y))$. For each property, we expect one result and allow to apply infinite number of transitions (denoted by the upper bounds $(1,\ast)$).

For Property $\text{prop1}$ CafeOBJ system returns $\text{true}$ for the first command, which means that a state satisfying the negation of $\text{prop1}$, namely a counterexample to $\text{prop1}$, has been found. This means that the property $\text{prop1}$ does not hold for $S_{core}$. Following the report of CafeOBJ, such a state (named as $s5$ here) can be reached from the initial state $\text{init}$ as follows:

$$\text{init} = [\text{register-i}]\Rightarrow s1 = [\text{register-j}]\Rightarrow s2 = [\text{create-ix}]\Rightarrow s3 = [\text{subscribe-jx}]\Rightarrow s4 = [\text{unregister-j}]\Rightarrow s5$$

The sequence of behaviors can be read as: a successor state is reached from its previous state by applying the state transition written in between them. For example, $s3$ is reached from $s2$ by member $i$ creating a channel $x$.

After checking the specification of action $\text{unregister}$, we can understand the cause of this unexpected sequence of behaviors: when unregister a member which has subscribed to a channel, the member’s name is not removed from the channel’s subscriber record (since the return value of observer $\text{channel}$ remains unchanged).

We now analyze what potential harm could possibly be caused by this problem. We know that, essentially, a member can send (receive) messages to (from) a channel only if
the member is a subscriber to the channel (see the precondition of action deliver). So, if the name of an unregistered member still exists in a channel that this member previously subscribed to, whether this may cause some unexpected behaviors of MSG-CORE? Let us see Figure 2.

The scenario illustrated in Figure 2 can be explained as: (1) a government agency A registers MSG-CORE using a name i, and another agency B registers MSG-CORE using a name j, (2) member i creates a channel x, and member j subscribes to channel x, (3) member i sends a message msg to member j through channel x, where j as one of the receivers and x as the intended channel are recorded in msg, (4) agency B unregisters member j, (5) an third government agency C registers MSG-CORE using the name j, (6) member j (representing agency C rather than agency B) receives msg from channel x. An assumption for this scenario is that a government agency can register MSG-CORE using any name as long as the name is currently not in use by other members.

The above scenario could actually be reported by CafeOBJ system when we try to check another property. The property named as Property 4 says that: if a member m received a message from a channel c and member m never unsubscribed channel c, then member m should have subscribed and still be subscribing to channel c. We omit the specification of the search command for Property 4 and the sequence of behaviors of MSG-CORE reported by CafeOBJ (same as the above six steps of the illustrated scenario).

For Property prop2
CafeOBJ returns nothing for the second command and keeps running. The reason for this is that even if we chose constants i, j, x and y to try to make the reachable state space of $S_{core}$ smaller, the restricted state space is still too large (actually still infinite) for CafeOBJ system to explore. However, if we set the upper bound for permitted number of applied transitions to 20, CafeOBJ system will return false, which means that within this upper bound, no states that satisfy the defined condition have been found.

One thing that we should keep in mind is that: even if CafeOBJ system returns false, since CafeOBJ system has not explored the entire reachable state space of $S_{core}$, we could not say that prop2 holds for $S_{core}$.

For Property prop3
CafeOBJ system returns true for the third command, which means that a state satisfying the negation of prop3, namely a counterexample to prop3, has been found. This means that the property prop3 does not hold for $S_{core}$. Following the report of CafeOBJ, such a state (named as s3 here) can be reached from the initial state init as follows:

```
init == [register-i] ==> s1 == [create-ix] ==> 
s2 == [unsubscribe-ix] ==> s3
```
We now analyze what potential harm could possibly be caused by this problem. Let us see Figure 3.

Figure 3 illustrates a scenario in which the extensions of MSG-CORE such as logging the messages passing through a channel, and encryption/decryption of messages, are employed. As illustrated, when the extensions are to be employed, the messages sent by member \( i \) through channel \( x \) (whose owner is member \( k \)) to member \( j \) are not directly sent to member \( j \), but firstly sent to member \( k \), and member \( k \) will enable those extensions. Therefore, if member \( k \) unsubscribes channel \( x \), then no extensions could be enabled.

**Verification of Revised \( S_{core} \)**

We give our suggested revisions to the framework MSG-CORE. To avoid the potential harm mentioned in Subsection 5.1.1, actually there are several possible ways, for example, (1) register the framework using fresh names, and disallow using a previously used name, (2) use extensions such as authentication, and (3) when unregister a member, remove the member’s name from the channel that this member has subscribed to. We choose the third one. In the CafeOBJ specifications for action `unregister`, the return value of observer `channel` is changed as follows:

\[
\text{ceq channel}(\text{unregister}(S,P),Y) = \begin{cases} 
\text{mkChn}(Y,\text{owner}(\text{channel}(S,Y)), \\
\text{del}(P,\text{subs}(\text{channel}(S,Y)))) & \text{if c-unregister}(S,P) \\
\text{channel}(S,Y) & \text{if c-unregister}(S,P) 
\end{cases}
\]

The revised specification says that: if the name of the to-be-unregistered member \( P \) exists in a channel \( Y \)'s subscriber record, then the name is removed from that record, and other components of the channel remain unchanged.

To avoid the potential harm mentioned in Subsection 5.1.3, we simply disallow the owner of a channel to unsubscribe the channel by changing the effective condition of action `unsubscribe` as follows:

\[
\text{eq c-unsubscribe}(S,P,X) = \begin{cases} 
P /\in \text{mSet}(\text{record}(S)) \text{ and } X /\in \text{cSet}(\text{record}(S)) \text{ and } \\
P /\in \text{subs}(\text{channel}(S,X)) \text{ and } \neg(\text{owner}(\text{channel}(S,X)) = P) 
\end{cases}
\]

Following the introduction in Subsection 3.3, we verify that properties \( \text{prop1} \), \( \text{prop2} \) and \( \text{prop3} \) are invariants of the revised specifications. Part of the proof score of \( \text{prop2} \) is described in this work as a demonstration example.
The basic formula to prove in each inductive case is declared in module ISTEP (which imports the module INV) as follows, where \( s \) and \( s' \) are constants of sort Sys (denoting an arbitrary state and its successor state) declared in INV:

\[
\text{op istep2 : Cname \to Bool eq istep2(C) = prop2(s,C) implies prop2(s',C)}.
\]

Let us consider the inductive case that action `unregister` preserves `prop2`. The case is split into 9 sub-cases based on the following 6 predicates:

- \( \text{bp1 } \triangleq q \in \text{mSet(record(s))} \)
- \( \text{bp2 } \triangleq \text{owns(member(s,q)) = empty} \)
- \( \text{bp3 } \triangleq q \in \text{subs(channel(s,c))} \)
- \( \text{bp4 } \triangleq c \in \text{cSet(record(s))} \)
- \( \text{bp5 } \triangleq \text{owner(channel(s,c)) \in mSet(record(s))} \)
- \( \text{bp6 } \triangleq \text{owner(channel(s,c)) = q} \)

where the constant \( q \) denotes an arbitrary member name, and \( c \) denotes an arbitrary channel name. The conjunction of \( \text{bp1} \) and \( \text{bp2} \) is the effective condition of action `unregister`.

The 9 sub-cases are as follows:

- **sub-case1** \( \triangleq \text{bp1} \land \text{bp2} \land \text{bp3} \land \neg \text{bp4} \)
- **sub-case2** \( \triangleq \text{bp1} \land \text{bp2} \land \text{bp3} \land \text{bp4} \land \text{bp5} \land \text{bp6} \)
- **sub-case3** \( \triangleq \text{bp1} \land \text{bp2} \land \text{bp3} \land \text{bp4} \land \text{bp5} \land \neg \text{bp6} \)
- **sub-case4** \( \triangleq \text{bp1} \land \text{bp2} \land \text{bp3} \land \text{bp4} \land \neg \text{bp5} \)
- **sub-case5** \( \triangleq \text{bp1} \land \text{bp2} \land \neg \text{bp3} \land \neg \text{bp4} \)
- **sub-case6** \( \triangleq \text{bp1} \land \text{bp2} \land \neg \text{bp3} \land \text{bp4} \land \neg \text{bp5} \)
- **sub-case7** \( \triangleq \text{bp1} \land \text{bp2} \land \neg \text{bp3} \land \text{bp4} \land \text{bp5} \land \text{bp6} \)
- **sub-case8** \( \triangleq \text{bp1} \land \text{bp2} \land \neg \text{bp3} \land \text{bp4} \land \text{bp5} \land \neg \text{bp6} \)
- **sub-case9** \( \triangleq \neg (\text{bp1} \land \text{bp2}) \)

Among the above sub-cases, sub-cases 2 and 7 need a predicate (named as `lemma1`) to strengthen the inductive hypothesis, and the remaining six sub-cases do not. We show the proof passage corresponding to sub-case 2:

```
open ISTEP -- arbitrary objects

op q : -> Mname .
-- assumptions characterizing a sub-case
-- eq c-unregister(s,q) = true .
-- eq q /in mSet(record(s)) = true .
-- eq owns(member(s,q)) = empty .
--
-- eq q /in subs(channel(s,c)) = true .
-- eq c /in cSet(record(s)) = true .
-- eq owner(channel(s,c)) /in mSet(record(s)) = true .
-- eq owner(channel(s,c)) = q .
-- successor state
-- eq s' = unregister(s,q) .
-- check if prop2 is true.
-- red lemma1(s,q,c) implies istep2(c) .

close
```

Constant \( c \) has been declared in the module INV that is imported by ISTEP. `lemma1(s,q,c)` is used in this passage to strengthen the inductive hypothesis `prop2(s,c)`. The proof
passage is prepared by human verifiers manually. When input this proof passage into CafeOBJ system, the result true is returned, which means that prop2 holds for this sub-case.

lemma1 is declared in module INV as follows:

\[
\text{op lemma1 : Sys Mname Cname -> Bool eq lemma1(S,P,C) = owner(channel(S,C)) = P implies C /in owns(member(S,P)) .}
\]

We need then to prove that lemma1 is an invariant of the revised \( S_{core} \). In this thesis, we omit this proof. In the following, we show some other properties of revised \( S_{core} \) that we have verified in our experiment. More detailed information of above work can be found in our under submission paper [78].

\[
\text{op addition-prop1 : Sys Mname Cname -> Bool eq addition-prop1(S,P,C) = P /in mSet(record(S)) and P /in subs(channel(S,C)) implies C /in cSet(record(S)) .}
\]

\[
\text{op addition-prop2 : Sys Mname Cname -> Bool eq addition-prop2(S,P,C) = P /in subs(channel(S,C)) implies P /in mSet(record(S)) .}
\]

\[
\text{op addition-prop3 : Sys Mname Cname -> Bool eq addition-prop3(S,P,C) = C /in owns(member(S,P)) implies owner(channel(S,C)) = P .}
\]

\[
\text{op addition-prop4 : Sys Mname Cname -> Bool eq addition-prop4(S,P,C) = not(C /in cSet(record(S))) implies not(C /in owns(member(S,P))) .}
\]

\[
\text{op addition-prop5 : Sys Mname Mname Cname -> Bool eq addition-prop5(S,P,Q,C) = (C /in owns(member(S,P)) and C /in owns(member(S,Q)) implies P = Q) .}
\]

\[
\text{op addition-prop6 : Sys Mname Cname -> Bool eq addition-prop6(S,P,C) = not(P /in mSet(record(S))) implies not(C /in owns(member(S,P))) .}
\]

2.4 Summary

We gave a brief background descriptions on domain engineering and formal method in this chapter.

By the domain engineering part, we showed how to apply domain engineering method into the public administration domain understanding in details. And by the formal method introduction, we first gave a brief background on the mainly formal tool we employed in this research – OTS/CafeOBJ method. Then, we described an experiment on using a specific formal method – the OTS/CafeOBJ method – to formally analyze a messaging framework proposed for providing citizens with seamless public services. In this experiment, two previously not well-clarified problems of the framework have been realized using the search command, and the potential harm that could possibly be caused by them is analyzed. We also suggested possible revisions to the framework, and proved by writing and executing proof scores that the revised specifications of the framework satisfies some desired properties. Based on this experiment, we advocated the use of formal methods, in particular formal analysis, to e-Government initiatives to help develop
reliable e-Government systems. In this case, falsification with the search command is fully automatic, and is used for finding logical errors by exploring part of the concerned system’s behaviors (therefore even if no errors are found, the correctness is not proved); verification by writing and executing proof scores needs much more human verifiers’ effort (for preparing the proof scores), and is used for proving correctness of the concerned system wrt desired properties by exploring its entire behaviors. Therefore there is a balance between the uses of the two techniques. Considering the fact that e-Government systems demand extremely high reliability, we believe that the effort paid for proving correctness is also worthwhile. For a more tight combination of the two techniques, we refer interested audience to [50] for its idea.

After given above preliminary knowledge, we start to discuss how to employ them in our research in the following three chapters.
Chapter 3

A General E-Government Model

There are two ways of constructing a software design. One way is to make it so simple that there are obviously no deficiencies. And the other way is to make it so complicated that there are no obvious deficiencies.

C.A.R. Hoare

In this chapter, we look more deeply into the public service domain. And present a simplified public service delivery model, called Generic Service Delivery Model (GSD model) based on the domain study and the real e-government system implementation work. The transparency discussion and case studies in the following chapter are mainly based on the GSD model in this chapter.

3.1 Formal Description of Public Administration Domain

3.1.1 Entities, Functions, Events and Behaviors

In Chap.2 gave a detailed sketch of the domain of public administration. Now we formalize this description. We go though the formal description corresponding to the domain understanding by given the formal description of entity, function, event and behavior respectively. Firstly, Entities: since we divide the Public Government into three part as: Lawmaker, Lawenforcer, Lawinterpreter and citizen who interacts with this three parts. We classify all those four kinds of entities as actor in the domain. And we illustrate actor by following CafeOBJ description:

mod! ACTOR {
  pr(EQL + NAME + ACTORSTATUS + ACTORTYPE + AUTHORIZATIONSET + OPERATIONSET )
  [Actor]

  op mk-actor : ActorName ActorStatus ActorType OperationSet AuthorizationSet -> Actor
  op actorname : Actor -> ActorName
  op actorstatus : Actor -> ActorStatus
  op actortype : Actor -> ActorType
  op operationrecord : Actor -> OperationSet
  op authorizationset : Actor -> AuthorizationSet
}
var AN : ActorName
var AS : ActorStatus
var AT : ActorType
var OS : OperationSet
var AUS : AuthorizationSet

eq actorname(mk-actor(AN, AS, AT, OS, AUS)) = AN.
eq actorstatus(mk-actor(AN, AS, AT, OS, AUS)) = AS.
eq actortype(mk-actor(AN, AS, AT, OS, AUS)) = AT.
eq operationrecord(mk-actor(AN, AS, AT, OS, AUS)) = OS.
eq authorizationset(mk-actor(AN, AS, AT, OS, AUS)) = AUS.

So the above codes describe entity actor as a sort mod! ACTOR and the actor consists with five components: name of the actor ActorName, the state of the actor ActorStatus, the type of the actor ActorType, the operations which have been done by the actor OperationSet, and the operations, which have been authorized to the actor, can be executed AuthorizationSet. And the operator actorname, actorstatus, operationrecord and authorizationset return the above components by given an concrete entity – actor.

As we mentioned in the previous paragraph, there are four kinds of actors, so we illustrate the three branches of government and citizen as following sort actortype:

mod! ACTORTYPE {
  pr(EQL)
  [ActorType]
  ops lawmaker lawinterpreter lawenforcer citizen : -> ActorType
  eq (lawmaker = lawinterpreter) = false.
  eq (lawmaker = lawenforcer) = false.
  eq (lawmaker = citizen) = false.
  eq (lawinterpreter = lawenforcer) = false.
  eq (lawinterpreter = citizen) = false.
  eq (lawenforcer = citizen) = false.
}

There is another important entity called Document in Public Government Domain. By different details we go into describe this entity, we can analyze different properties in PA Domain. Here we in focus on the structure of PA Domain so the method we formalize document as same as Actor. The entity document DOCUMENT is constructed document name DocName, document creator ActorName, document state DocumentStatus, document type DocumentType, content of this document Information, the operation have been applied on this document OperationList and the handover of the operation right on this document between actors authrecord.

mod! DOCUMENT {
  pr(EQL + DOCUMENTSTATUS + NAME + DOCUMENTTYPE +
      INFORMATION + AUTHORIZATIONLIST + OPERATIONLIST)
  [Document]
  op mk-document : DocName ActorName DocumentStatus DocumentType
      Information OperationList AuthorizationList -> Document
  op _=_ : Document Document -> Bool
So six concrete operations which often be applied on public administration documents have been described in our formal description as follows:

```
mod! OPERATION {
  pr(EQL)
  [Operation]
  ops create edit read copy calculate shred : -> Operation
}
```

here refers to six kinds of document operations may be engaged. *create* refers to a document creation, *edit* means edit the document, *read* means to read the document, *copy* means make a copy on this document, *calculate* refers to do calculation based on the document information and *shred* means to shred the document in the document set of this public administration systems. And the document states are be described as the same way.

We lift two main function in public administration domain, are *operate* and *authorize* as we mentioned in previous chapter. Both of them are close related to document. We employ pair to record who applied operation and what the operation does. And we use three tuple to record the handover of the operation right on the document. The data structure of pair and three tuple are defined as following:

```
-- used for describing document trace
-- and operation record
mod! PAIR(M :: TRIV, N :: TRIV) {
  pr(EQL)
  [Pair]
  op [...] : Elt.M Elt.N -> Pair
  op left : Pair -> Elt.M
  op right : Pair -> Elt.N
  var P : Pair
  vars X1 X2 : Elt.M
  vars Y1 Y2 : Elt.N
  eq left([X1, Y1]) = X1 .
  eq right([X1, Y1]) = Y1 .
  eq ([X1, Y1] = [X2, Y2]) = (X1 = X2) and (Y1 = Y2) .
}
```

```
-- used for describing actor authorization
-- and document authorization record
mod! TPAIR( M :: TRIV, N :: TRIV, O :: TRIV ) {
  pr(EQL)
  [TPair]
```
With above base data types in public administration domain, we can describe the public administration system as well as the functions in it. The whole public administration system is considered as an OTS and we illustrate behavioral module of the system by figure 3.1.

Figure 3.1: Illustration of Behavioral module of Public Administration System

We regard Public Administration System as an observer transaction system. Formalize it as an hidden sort called \( \text{PA} \). There are two observers to observe the entities states in this systems, they are \textit{actor} and \textit{document}. The \textbf{functions} of the system are written as transition actions in the OTS. They are \textit{authorize} and \textit{operate}. Here is the signature of the systems being described by CafeOBJ.

```plaintext
mod* PA {
    pr (ACTOR + DOCUMENT + OPERATION)
    pr (EQL)
    *[System]*
    -- any initial state
    op init : -> System
    -- observations in PA System
    bop actor : System ActorName -> Actor
    bop document : System DocName -> Document
    -- actions
    bop authorize : System ActorName ActorName
}```
The “init” presents the initial state of the public administration system. We assume that in the initial state of the system, an arbitrary actor who named A, its state is exiting, type is citizen, and the operation record and authorization record sets of this actor (A) are empty. And an arbitrary document named D in the initial state, is law document (lmdoc) which content is some commonsense and is proposed by committee members (committee). The record list of operation and authorization of operation right on the document are all empty. The following codes illustrate our above narrative description.

-- initial state
eq actor(init, A) = mk-actor(A, exiting, citizen, em-os, em-au) .
eq document(init, D) = mk-document(D, committee, idled, lmdoc, commonsense, nil-os, nil-au) .

And given the description of observer actor: “bop actor : System ActorName -¿ Actor”. It refers to give a particular system state (System) and actor’s name (ActorName), the observer actor will return the actor state (actor) in this system state. The description of observer “document” returns document state as the same way. The two actions “authorize” and “operate” change the system state of “PA”. “authorize” changes actor’s “AuthorizationSet”, which records the operation permission to the document of this actor; and document’s “AuthorizationList”, which records the handover of the operation right on document.

And the effective conditions of above actions can be understand as event. While something happened, if it fulfills the effective conditions of the actions, then the actions will be applied. The effective conditions are defined by following equations in CafeOBJ.

-- effective condition of authorize
op c-authorize : System ActorName ActorName DocName Operation -> Bool
eq c-authorize(S, A1, A2, D, O) = actorstatus(actor(S, A1)) = exiting
and actorstatus(actor(S, A2)) = exiting
and ((A1 = doccreator(document(S,D)))
or ( < D, O, A1 > /in authorizationset(actor(S, A1)))).

-- effective condition of operate
op c-operate : System ActorName DocName Operation -> Bool
eq c-operate(S, A, D, O) = < D, O, A > /in authorizationset(actor(S, A))
and (docstate(document(S, D)) = idled) or (O = creat).

Taking the condition of “authorize” as an example. By given a system state, an actor’s name who authorizes the operation, and another actor’s name who be authorized, the document name which operation right on it be handed over, and the operation which can be applied on document; the operator “c-authorize” returns a boolean value true/false depending on the condition hold or not. And the equation comes after it describes the conditions of applying authorize are: (i) two actors who are involved should be exiting from document handling; (ii) the actor who authorize the operation right of the document should be the creator of the document, otherwise, the actor should have corresponding operation right on this document. And the following CafeOBJ equations define the state changes after the actions.
-- authorize

cqe actor(authorize(S, A1, A2, D, O), A1)
  = (if (not(A1 = A) and not(A2 = A)) then actor(S, A)
    else if(A = A2) then mk-actor(A2, exiting, actortype(actor(S, A2)),
        operationrecord(actor(S, A2)),
        authorizationset(actor(S, A2)))
    else mk-actor(A1, exiting, actortype(actor(S, A1)),
        operationrecord(actor(S, A1)),
        authorizationset(actor(S, A1))) fi fi)
  if c-authorize(S, A1, A2, D, O) .

cqe document(authorize(S, A1, A2, D, O), D1)
  = (if not(D = D1) then document(S, D)
    else mk-document(D, doccreator(document(S, D)),
        idled, doctype(document(S, D)), docinfo(document(S, D)),
        doctrace(document(S, D)), add(< D, O, A1 >, authrecord(document(S, D)))) fi)
  if c-authorize(S, A1, A2, D, O) .

cqe authorize(S, A1, A2, D, O) = S if not c-authorize(S, A1, A2, D, O) .

-- operate

cqe actor(operate(S, A, D, O), A1)
  = (if not(A1 = A) then actor(S, A)
    else mk-actor(A1, operating, actortype(actor(S, A1)),
        operationrecord(actor(S, A1)),
        authorizationset(actor(S, A1))) fi)
  if c-operate(S, A, D, O) .

cqe document(operate(S, A, D, O), D1)
  = (if not(D = D1) then document(S, D)
    else mk-document(D, doccreator(document(S, D)),
        idled, doctype(document(S, D)), docinfo(document(S, D)),
        doctrace(document(S, D)), add([A, O], authrecord(document(S, D)))) fi)
  if c-operate(S, A, D, O) .

cqe operate(S, A, D, O) = S if not c-operate(S, A, D, O) .

Take the action “operate” for instant; the above equations have the following meaning. (i) The first equation means, if the effective condition of action “operation” held and “operate” action happened in the public administration domain, the observer actor can observe the arbitrary actor who named A1, if the A1 is the one who apply the action “operate” (namely, A), then actor’s operation record (operationrecord) will record this action (plays which operation O on document D, illustrated by [D, O]) and other components of the actor A keeps same. And if the actor A1 is not the one who apply the action “operate”, then the actor state keeps same. (ii) The second equation refers to the document state changes if the document is exactly which the operation be applied, and the document state keeps same if the document is not the one be operated; while the effective condition of action “operate” held. (iii) The third equation shows the system state will keep same, if the effective condition of the action does not hold. You can get complete CafeOBJ specification from appendix. And a CafeOBJ comment starts with – and terminates at the end of the line.

So the behavior of the domain can be considered as a sequence of transition actions applied on public administration system.

open PA
ops applicant officer : -> ActorName .
op application : -> DocName .
op s : -> System .

40
The above CafeOBJ code describes a behavior in public administration domain. The “open” command open the module “PA” which is described at the previous part of this section. We suppose there are two actors who named applicant and officer, and a document which named “application” here. Given an arbitrary state “s”, a applicant start a service application, which made the condition of the action operate holds (“operate(init, applicant, application,create)”), can be viewed as an event in this case. Therefore, this event triggers an action sequence: firstly, applicant create an application in the initial state; secondly, the applicant submit this application, this implies applicant authorize the read and edit right on the document application to the actor officer; at last the officer edit the application by writing their decision. All of above actions can be considered as description of behavior in the domain. And many other kinds of events and behaviors can be described following the same way. So till now, we formally described all the four aspects of domain understanding: entities, functions, events and behaviors in chapter 2. In the coming subsection, we are going to discuss the usage of above formal documents.

3.1.2 Simple Reasoning based on Domain Description

Though our formal description of public administration domain is still standing at a high-level point of view, the formal document already has its advantage for our study.

On one hand, the formal document can help us to understand the requirement from engineering side. As we discussed in the preliminary part, the formal specification can helps us recognize the role of the early phases of the development process, namely requirement analysis, specification and high-level design. This is because, as with any specification, formal specification results from the requirement analysis and forms the basis for the subsequent design. These early phases acquire such importance, on the one hand, because of the progressively increasing costs of remedial work as the development process advances and, on the other, because of the highly error-prone nature of the activities involved in these phases. There is also the possibility that the importance of these phases will be underestimated by the developers and, as a result, they will receive low priority because of commercial pressures or the lack of trained specialists.

On the other hand, based on the formal description, we can do some simple reasoning. For example, by giving an action sequence, we can query if the action is allowed in the current system state. The CafeOBJ description are shown as follows:

\[
\text{red } \langle \text{decision, edit, police} \rangle \\
\quad \text{in } \text{authorizationset(actor(authorize(operate(init, police, decision, create), police, citizen, decision, read)), police)}.
\]

As shown in above code, supporting there are two actors “police” and “citizen” in this case. police create a decision document, because the citizen broke the traffic rule. After the police authorizes the citizen to read the decision document. After above two
actions applied, we query in this system state, if the police has right to edit the decision document. The CafeOBJ system will return a boolean value (True/False) depending on the operation is allowed or not.

Even more, by giving different initial state and adding more concrete information description of the domain, the module can refer to different cases, such as: employee training, public service delivery and so on. And by more detailed formal description of the domain, some more interesting properties are able to be described and mathematically proved; this is also what we are going to do in the rest of this research.

3.2 Formal Description of General Service Delivery Model

3.2.1 Introduction of Public Service Delivery

After domain understanding, according to the relationship between e-government and public administration, which we have mentioned in Chapter 2, we move our focus on describing e-government system and its’ properties. Since the current e-government system are mainly for delivering public service, the following model dedicates on presenting that one of the three government branches, namely law enforcer, interacts with citizens and delivers public service. Following the style of domain understanding, the General Service Delivery (GSD) model contains entities: actors and document. Still with actor citizen, the Actors law enforcer are divided into more concrete types, such processing officer, collaborative agency and so on. And the actions in the system are lifted up as the following five steps for government providing public services (as shown in figure 3.2):

1. Apply public service; in this step, citizen who want to apply public service by filling corresponding application form

2. Evaluate application; officer of the public agency evaluate citizen application after receiving the request, and make decision or return the application by asking more supplementary documents.

3. Confirm evaluation; a different officer in this agency confirms the evaluation result and decide if agree with the decision or give back to re-evaluate.

4. Notify decision; public agency notifies the service applicant about the result of his/her application.

5. Execute decision; an officer or a public agency is supposed to execute the decision, if the application was accepted.

3.2.2 Basic Data Type

Before introducing the OTS model of the generic workflow system, we first describe some key data types to be used in the OTS, which include: document, actor, operation and
authorization. Each document (Doc$^1$) in the welfare system is constructed using the CafeOBJ operator \texttt{mk-doc} that takes the following eight arguments:

1. Document name (\texttt{Dname}): the name of the document. This component is the identifier of a document.

2. Document type (\texttt{Dtype}): we classified the documents into three types: law document (\texttt{ldoc}$^2$), which can be created only by law making branch of government, based on decision and history record documents of government; decision document (\texttt{ddoc}), which can be created only by law enforcing and interpreting branches of government, based on citizen’s application; the private document (\texttt{pdoc}), such as application, which can be created by every citizen, and government can only process the document but cannot pass it to other citizen without the applicant’s authorization.

3. Document creator (\texttt{Actor}): the creator of the document.

4. Document information (\texttt{Info}): the current information of the document.

5. Document trace (\texttt{Dtrace}): a list of pair (of actor and operation, which is to be introduced latter) to record the document operation history.

6. Refer document (\texttt{Doc}): the document based on which another document is created. For example, application form is generally created based on administrative law document.

---

$^1$As a general rule of CafeOBJ specification, we use words with initial capital characters, such as \texttt{Sort}, to denote the sort names of corresponding data types.

$^2$We use CafeOBJ constants to denote such specific types of documents.
7. Document label (Dlabel): possible labels of a document are: applied, evaluated, confirmed, notified, accepted, unsatisfied, and executed. Each of them denotes a workflow processing stage as mentioned in Section 3.3.1.

8. Document Authorization (DocAuth): we use a authorization (Authorization) list to record who (+Aname+) authorized what kind of operation (Operation) right to what kind of actor role (ActorRoleSet).

Given a document in the form \( mk\text{-doc}(N:\text{Dname}, T:\text{Dtype}, DC:\text{Actor}, DI:\text{Info}, DT:\text{Dtrace}, R:\text{Doc}, DL:\text{Dlabel}, DA:\text{DocAuth}) \), projection operators docName, docType, docStatus, docCreator, docInfo, docTrace, docRefer and docLabel docAuth are also defined to obtain each of its components.

Each actor in the system is constructed in the similar way. It is constructed using the CafeOBJ operator \( mk\text{-actor} \) that takes the following five arguments:

1. Actor name (ActorName): the name of the actor. This component is the identifier of an actor.

2. Actor status (ActorStatus): the status of actors. Possible status of an actor is: handling and idling. If any actor of the system is in the status handling, it means that the document processing has not been completed.

3. Actor knowledge (DocNameSet): a set of documents that an actor has right to study on them. For example, a citizen has right to learn basic law, therefore basic law document supposed in actor knowledge.

4. Actor trace (DocTrace): an actor can trace the document state which created by him/her.

Given an actor in the form \( mk\text{-actor}(N:\text{Aname}, AS:\text{Astatus}, T:\text{Atype}, AP:\text{Opset}, AO:\text{Opset}) \), projection operators aname, astatus, atype, aright and aoblit are also defined to obtain each of its components.

We abstract six basic operations (of sort Op) from the real e-Government systems that can be performed on documents. Constructors of the six operators (which takes a document name, an actor name etc as arguments) are: create, edit, read, copy, shred and authorization. We use the predicates iscreate, isedit, isread, iscopy, isshred and isauth to check the attribution of an operation. Given different kinds of operations, projection operators are defined as follows: docofo returns the document on which the operation performs; actofo returns the actor that can/should perform this operation; infoofo and editofo return the information created and edited by the operation, respectively; timeofo returns how many times the operation can be performed on this document by the actor; typeofo returns the authorization kinds of the operation (we use permission to denote that the operation is one that can be performed, and obligation to denote that the operation is one that must be performed on the document by the actor; attiofo returns the attitude of the actor on this document (we defined two attitudes in this model: agree and notagree (of sort Atit)).

Operation set is a set of operations. We can add (add) into, subtract (sub) from, and check if an element (_in_) is in this set, and can perform set operations: union (_U_), intersection (__&_) and complementation (not).

The keyword mod! indicates that the module is a tight semantics declaration, meaning the smallest model (implementation) that respects all requirements written in the module.
The contents of the module are enclosed in the keywords \{ \}. The keyword `pr` is used to import a module, here the module `OPERATION` describing data type `Op`. The key words `op` and `eq` are used to define the operators and equations of this module.

### 3.2.3 OTS Model and Specification

The OTS model of the generic workflow system is defined in a CafeOBJ module with the name `PG` using the CafeOBJ keyword `mod*`, which indicates that the module is a loose semantics declaration, meaning an arbitrary model (implementation) that respects all requirements written in the module. The module `PG` imports all the data type modules defined in advance. A hidden sort `Sys` is declared in the module as `*[Sys]*` by enclosing it with `*[ and ]*`, which denotes the universal state space of the OTS model.

In module `PG`, two observers denoted by CafeOBJ observation operators `actor` and `doc` are declared as follows:

```
  bop actor : Sys ActorName -> Actor
  bop doc : Sys DocName -> Doc
```

Given a state (say \( s \)) of the OTS and an actor’s name (say \( an \)) and a document name (say \( dn \)), observers `citi`, `lawmaker`, `lawenfor` and `lawinter` return respectively the status (value of each component) of the citizen, lawmaker, lawenforcer and lawinterpreter with the name \( an \) and being working on the document \( dn \) in the state \( s \). And given a state of the OTS, observer `doc` returns the status (value of each component) of the document with the name \( dn \).

A constant `init` is declared as “`op init : -> Sys`” to denote any initial state of the OTS model of the PA system. The initial state is characterized by the following equations:

```
-- [0] equations of initial state
eq actor(init,C) = mk-actor(C,idling,basicLawSet,ndt).

eq doc(init,D) = (if D /in basicLawSet
                    then basicLaw else emptyDoc fi) .
```

In the initial state of the PA system, we suppose that there are some basic law documents which have been published, and both citizen and administrative officers should start the public administration procedure based on these basic documents. In the first equation, variable \( C \) denotes an arbitrary actor’s name(note that the name component is the identifiers of both actors and documents). The right-hand side of the equation describes the components of actor with the name \( C \). The actor components are composed using the operator `mk-actor`. Any actor denoted by \( C \) is initially in the status `idling`, and have knowledge with some basic law `basicLawSet`. The second equation says that at the initial stage of this system, there are some basic law documents, if a document named \( D \) are not in basic law document set then this document should be not existed `basicLawSet`.

Based on the narrative description in Section 3.2.1., six transitions, which characterize the processing of documents and track document, are declared as follows:
Table 3.1: Equation descriptions of “evaluate” action in the GSD model

\[
\begin{align*}
\text{ceq actor(evaluate(S,C,D,I),C1)} &= \begin{cases} 
\text{actor(S,C1)} & \text{if } \text{not(C1 = C)} \\
\text{mk-actor(C,handling,actorKnow(actor(S,C)), actorTrack(actor(S,C)))} & \text{else}
\end{cases} \\
\text{if c-evaluate(S,C,D,I)}.
\end{align*}
\]

\[
\begin{align*}
\text{ceq doc(evaluate(S,C,D,I),D1)} &= \begin{cases} 
\text{doc(S,D1)} & \text{if } \text{not(D1 = D)} \\
\text{mk-doc(D,docCreator(doc(S,D)),docStatus(doc(S,D)),docType(doc(S,D))},
\text{add([C,evaluated],docTrace(doc(S,D))),
\text{del(auth(left(getTerm(submitSD,docTrace(doc(S,D))))),evaluated,agenciesSet),
\text{(if I = noenough then docAuth(doc(S,D))
\text{else add(auth(C,decide,agenciesSet),docAuth(doc(S,D)))) fi)),
\text{(actorKnow(actor(S,C)) U docRefer(doc(S,D))), (if I = noenough}
\text{then submited else evaluated fi)) fi}) & \text{if } c\text{-evaluate(S,C,D,I)}.
\end{cases}
\end{align*}
\]

\[
\begin{align*}
\text{ceq evaluate(S,C,D,I) = S} & \text{if not c-evaluate(S,C,D,I)}. 
\end{align*}
\]

Take transition denoted by the CafeOBJ action operator evaluate as an example. It characterizes that actor who belongs to public agency role agenciesSet evaluates an application document created by a citizen, and transfers the evaluated application document to another officer. The effective condition of the transition is as follows:

\[
\begin{align*}
\text{op c-evaluate :Sys ActorName DocName Info -> Bool} \\
\text{eq c-evaluate(S,C,D,I) = C /in agenciesSet and}
\text{not (C = docCreator(doc(S,D))) and}
\text{docType(doc(S,D)) = pdoc and}
\text{docLabel(doc(S,D)) = submited and}
\text{(auth(docCreator(doc(S,D)),evaluate,agenciesSet) /in docAuth(doc(S,D)))}. 
\end{align*}
\]

The condition denoted by c-evaluate says: there is an application document (D) in the status submited; the actor (C) who is going to evaluate the application document is an officer of public agency (agenciesSet); actor C is not the creator of D; and his/her role has right to perform this evaluation.

The CafeOBJ specification describing execution of transition evaluate is shown in Table 1. If evaluate is applied when the condition holds: (1) The state of actor who apply this action evaluate change into handling and the knowledge and track about his own document of this actor do not change. (2) The document which be evaluated, will
record the evaluation information docInfo, who evaluated it in document trace, delete the evaluation right for this role and authorize other role for next processing step. (3) If evaluate is applied when the effective condition does not hold, nothing changes.

The remaining transitions can be defined in a similar way, which are omitted here.

Based on this model, some simple reduction can be occurred, such as: query if a citizen can trace a document. By inputting the following CafeOBJ code, the CafeOBJ system can return ‘true” or false, depending on the citizen’s right on tracing the document.

```
open PA
op a : -> ActorName .
op d : -> DocName .
op s : -> Sys .
red (docCreator(doc(s,d)) = a) implies c-track(s,a,d) .
red docLabel(basicLaw) = issued and c-track(s,a,docName(basicLaw)) .
close
```

The above paragraph uses CafeOBJ command reduce (“red”) to query if the creator of the document can trace the document he created by the first “red” sentence. And the second reduce command query if the citizen can trace the document, which is a law document which has been issued to public.

### 3.3 Summary

In this section, we proposed a Generic Service Delivery Model based on our understanding of public administration domain. The domain analysis guarantee the model contains the important aspects of the area as well as the formal model based on domain understanding can refine some interesting properties for further formal discussion (such as, transparency). Based on the formal description some desired properties such as transparency can be described. For transparency discussion, there are some interesting issues can be addressed in the following chapters, such as:

- What kind of transparency this GSD model is fulfilled?
- Which transparency degree this GSD model can achieved?
- For different actors in this GSD model, are the transparency degrees same to them? And etc.

To study above questions, we start the research in Chapter 4.
Chapter 4

Study on Government Transparency

No-one would dare say that they were against transparency (...): It would be like saying you were against motherhood or apple pie.

Joseph Stiglitz

As the above quote highlights most people engaged in government and public administration systems claim to be in favor of government transparency. However there is still a controversy in the recent academic literature and in the newspapers about the transparency of government. In fact, almost no one involved in the political or administrative discussion about the transparency actually denies that transparency is socially beneficial. More and more governments realize and claim to be open and transparent.

So how the controversial debate about transparency come from? Despite the seemingly wide-spread consensus that transparency is desirable, is that people have different views as to what government transparency is and there is no uniform methodology to gauge it.

In this chapter, it starts from the literature overview of transparency, and tries to provide a panoramic picture around the study of transparency and its related concepts. Then, focus on the transparency issues in government systems, especially on e-Government systems to discuss the precise definitions, categories, its related factors and evaluation methods of transparency. After fully understand the transparency, we connect narrative (informal) descriptions around transparency with formal descriptions (CafeOBJ descriptions), define and study transparency from the formal view based on the model which we introduced in the previous chapter. At last, summarize the transparency studies in this section.

4.1 Literature Overview

Although being widely used, transparency has been rarely defined in a precise way. one of the difficulties of doing this is partially due to transparency is used in so many different fields. In natural science, transparency is used for describing materials property of condition for transmitting light to the outside or vice versa. In public administration field, transparency refers that information is freely available and directly accessible to those who will be affected by decisions and that enough information is provided in easily

\footnote{1Quoted in Financial times 5 October 1998.}
understandable forms and media.

Even there has some common knowledge on the concepts of transparency, different researchers still have different understanding depending on their ways and areas to discuss it [80]. In [13], transparency has been understood as constituting a laymans basic map of the organization as depicted in the information on the site and reveals the depth of access it allows, the depths of knowledge about processes it is willing to reveal, and the level of attention to citizen response it provides, in his evaluation on the citizen satisfaction of government web-service. In tracking financial resource and the identification of process features, transparency has been identical as openness of financial allocations, cost estimates and expenditures to stay in the public domain [7]. In government fiscal study, transparency be quantitatively transferred into the share of the control-error variance arising from the observed component, under which the public perfectly infers the banks intention each period. The process should be transparent so that the public can see what is going on and how decisions are being made [35].

4.1.1 Narrative Description of Transparency

E-Government is a ground-breaking domain, therefore, before discussing the transparency in government systems; we have a glance of panorama picture on transparency concept in different areas, and then combine them together for servicing government transparency discussions. In different area, the descriptions of transparency may difference, we firstly distinguish them from the following areas.

- physical domain: In physical domain, transparency may refer to: Transparency (optics), the physical property of allowing the transmission of light through a material.

- humanity domain: Transparency, as used in the humanities, when used in a social context, implies openness, communication, and accountability. It is a metaphorical extension of the meaning a "transparent" object is one that can be seen through. Transparent procedures include open meetings, financial disclosure statements, the freedom of information legislation, budgetary review, audits, and so on.

- Economic domain: In economics, a market is transparent is described as, if much is known by many about: (a) what products, services or capital assets are available. (b) what price. (c)where does the transaction occur. Considering the market transparency is one kind of information transparency. Price, as one of the important information in market, there are usually considered as two types of price transparency: (1) customers know what price will be charged to them, and (2) customers know what price will be charged to other customers [55].

- literal domain: In literal domain, or called linguistic transparency is a phrase which is used in multiple, overlapping subjects in the fields of linguistics and the philosophy of language. It has both normative and descriptive senses.

- Political domain: In politics transparency is introduced as a means of holding public officials accountable and fighting corruption. When government meetings are open to the press and the public, when budgets and financial statements may be reviewed by anyone, when laws, rules and decisions are open to discussion, they are seen as
transparent and there is less opportunity for the authorities to abuse the system in their own interest.

- **Information Technology domain**: In the area of information technology, the transparency may refer to Information transparency as well as design transparency (user and engineering design considerations). Information transparency describe the accessability of information itself, which closely connected with classified information. And the design transparency, which can be divided into many kinds, for instance: location transparency if the names used to identify network resources are independent of both the user’s location and the resource location; and network transparency if there is no difference between the centralized database and the distributed database. More detailed classification are presented in the next subsection.

- **Philosophy domain**: Some scholars even tried to discuss transparency from philosophy aspect of view. They described transparency as a property of epistemic states defined as follows [58]:

  An epistemic state E is weakly transparent to a subject S if and only if when S is in state E, S can know that S is in state E;

  An epistemic state E is strongly transparent to a subject S if and only if when S is in state E, S can know that S is in state E, AND when S is not in state E, S can know S is not in state E.

  And here is an example: pain is usually considered to be strongly transparent: when someone is in pain, he knows immediately that he is in pain, and if he is not in pain, he will know he is not.

To illustrate transparency more clearly, further classification in some of above area are shown in the next subsection.

### 4.1.2 Categories of Transparency

In literal uses:

- Electromagnetically induced transparency, an effect in which a medium that is normally opaque at a particular wavelength is caused to become temporarily transparent

- Pentimento, an alteration in a painting, often revealed with growing transparency in paints with age

- Transparency (photography), a still, positive image created on a transparent base using photochemical means

- Transparency (projection), a thin sheet of transparent material for use with an overhead projector

In computing and mathematics:

- Transparency (computing), user and engineering design considerations, including Location transparency if the names used to identify network resources are independent of both the user’s location and the resource location Network transparency if there is no difference between the centralized database and the distributed database
- Transparency (data compression), the ideal result of lossy data compression
- Transparency (telecommunication), the property by which a transmission system passes a signal through without changing its form or content
- Transparency (graphic), for overlay and translucency in PNG, GIF, and TIFF files
- Transparency (pseudo), or background translucency in the X or X11 Window System
- Referential transparency in programming designates a deterministic function
- Security through transparency, a security engineering methodology which attempts to get as many people as possible reviewing code, with the hope that they will maintain or at least point out bugs in it, improving security over time.

In humanities and business:
- Media transparency, in the communications industry
- Radical transparency in management
- Transparency (behavior), a metaphor implying visibility in contexts related to the behavior of individuals or groups
- Transparency (linguistic), a term used in linguistics and the philosophy of language
- Transparency (market), a term in economics
- Transparency (philosophy), an adjective applied to a state in which the subject can be aware of being in that state

And there are many other social organizations which involved in social transparency issues, such as: Transparency International, an organization working for governance, corporate, banking and association; transparency Transparency (Guatemala), a political party in Guatemala (Transparencia).

### 4.1.3 A Brief Overview

We are concerned about transparency because it enables two interlocked engines of welfare and development, namely governance and economic markets. More openness and information sharing enables the public to make informed political decisions, improves the accountability of governments, and reduces the scope for corruption. Greater transparency is also essential to the economy: it improves resource allocation, enhances efficiency and increases growth prospects. Information imperfections in markets increase transaction costs and give rise to market failures. Though market failures hamper the working of all markets, they especially affect capital markets. In the recent financial crises literature, several references\(^2\) are made to “lack of transparency” as one of the factors that either caused or contributed to the prolonged crises.

Despite this, today even most ostensibly open and democratic societies lack sufficient transparency in governance and finance. As argued by Stiglitz [38] there is little reason

\(^2\)Such references are primarily in popular magazines and newspaper articles.
for this. Arguments against more transparency, while merited in a few instances, are often not only limited in application, but simply wrong. This essay departs from this observation, elaborating the means of studying and implementing transparency.

Despite the perceived importance of transparency there has been little theoretical or empirical effort to study it. For example its role in enhancing long-run growth prospects or improving stability of markets and averting short/medium term crises. In both former respects complex questions and challenges present themselves. It has been argued that increased transparency may improve financial stability and reduce market volatility. Yet, conceptual work suggests that increased transparency need not be welfare enhancing and furthermore may increase market volatility. Understanding the conditions under which more transparency may improve or worsen outcomes will therefore refine our knowledge and will help shape better policy.

Empirical work is lacking partly because of data problems. These arise from the difficulty of measuring “transparency” given that it deals with agents who are hiding information. Therefore a key empirical challenge is to define a measure of transparency that is empirically tractable. Such an exercise will highlight the requirements of the data as well as enable us to assess its determinants and evaluate its impact on the outcomes of interest.

Some researchers are also believe that a working understanding of transparency should encompass the following attributes: access, comprehensiveness, relevance, and quality and reliability. They are detailed below:

Access-- Laws and regulations ensure, at least in principle, that information remains available to all. But information must also be accessible. In part, this is aided by the institutions and venues that facilitate its flow. They include media such as newspaper, radio, TV, public information notices, the Internet, and word of mouth. Lack of education is detrimental to transparency - it limits the ability of an individual to access, interpret and use information. Strong equity considerations attend the need for access. Information should be accessible to all on equal terms. However, it is often profitable to delay or limit access to useful information, in which case access becomes hostage to ability to pay. Therefore there is a need to enforce timely, equitable dissemination of information.

Relevance – Information must be relevant. Ensuring this is difficult, first because information is subjective; depositors need information to ensure safety of deposits; investors need information about liabilities and risks; and the public about current economic conditions, policies, and so forth. Second as sources of information such as the Internet proliferate, paradoxically information overload threatens to dilute relevance.

Quality and Reliability – Information should be of good quality and reliable, timely, complete, fair, consistent and represented in clear and simple terms. Standards for quality must be ensured, possibly through verification by external agencies or auditors or standard setting organizations. Consistency in the use of processes to obtain information and in the formats of information disseminated ensures comparability and so allows assessment of changes over time. The criteria and methodologies used to assess information, as well as changes in such methodologies, should be fully disclosed. Such measures are an important way of preventing deliberate withholding or distortion of information - lying. Dishonest reporting is deterred by the presence of various “watchdog” institutions ranging from professional accountants or agencies, credit bureaus, an independent press, stakeholder

\[^{3}\text{Refers to the information economics literature, and literature on information revealed by prices in rational expectation models.}\]
feedback, to even academic researchers. Furthermore, assuring quality and reliability often requires going beyond integrity: rigorously evaluating signal-to-noise ratios in a piece of data is often a methodological and empirical challenge even for institutions and individuals of the highest standards of probity.

By measuring transparency, these various aspects of transparency invoke specific policies and institutional arrangements. In order to evaluate these policies, we must be able to measure transparency C a difficult task in part because of the complex understanding of transparency that we have adopted. Conceptually, a statistical measure of transparency is the precision of the information that is obtained, in turn a function of its relevance and finally, its quality. “Lack of transparency” in the case of accounting information for example, may be measured by comparing officially disclosed balance sheet information with the assessments of auditing agencies that investigate firms for credit approval. Highly transparent firms will have little discrepancy between the officially disclosed information and that assessed by auditors. An important prerequisite for such measurement is that the data of such evaluations is accessible.

Recent attempts to measure transparency have used proxies such as “weak rule of law” and “corruption” that are associated with lack of transparency but do not fully reflect all the above considerations. A refinement of this approach is to formulate an index using proxies for the characteristics outlined above. However, a serious impediment to measuring transparency is poor data quality- detailed information on publicly disclosed information, the various disclosure standards, evaluations by independent auditors of the categories of information disclosed. With improved data, one can systematically measure transparency, identify its determinants, and quantify its impact on the relevant economic variables. An attempt to measure accounting transparency using data from Indian firms and assessing its impact on investment activity is currently underway [54].

Before proceeding from measuring transparency to implementing it, it is necessary to assess if transparency always desirable. Proscriptions against disclosure abound in all societies. Is there sense in these proscriptions?

As argued by Stiglitz [38], on the whole societies’ preference should be in favor of greater openness and transparency. Conceptually, the information economics literature supports the notion that better information will improve resource allocation and efficiency in an economy. Disclosure of financial information directs capital to its most productive uses, leading to efficiency and growth. Lack of transparency can be costly, in both political and economic terms. It is politically debilitating because it dilutes the ability of the democratic system to judge and correct government policy, cloaking the activities of special interests, and creating rents by giving those with information something to trade. The economic costs of secrecy are equally staggering, affecting not only aggregate output but also the distribution of benefits and risks in the economy. The most significant cost is that of corruption, which has a documented adverse effect on investment and economic growth.

Arguments against more transparency while merited in a few instances, are often not only limited in application, but fundamentally flawed. There is arguably some merit in more philosophical rights-based exceptions on the grounds of privacy and confidentiality.

4See Kane:1999, for specific details in the context of financial information
5There is a growing literature on the relation between corruption and growth in particular, initiated by Mauro (1995). More broadly for a number of governance dimensions, see Kaufmann, Kraay and Zoido-Lobaton (“Governance Matterss”, 1999b).
Even these however need to counter not only the instrumental benefits of transparency, but also powerful arguments about the intrinsic rights of citizens to know. More dubious are exceptions made on grounds such as national security, stability, non-interference in delicate negotiations, or deference to public unity. Notably, some possible exceptions on the grounds of ensuring stability in financial markets are treated in greater detail in the next section. To the extent that such appeals are made, they need to be highly limited, and the limits exposed to public debate. Particular scrutiny should be directed at invocations of confidentiality, market stability or national security.

Research to inform such debates is currently inadequate, failing to qualify arguments both for and against transparency. Most concretely, this pertains to debates on the need for financial reform. For example, arguments about the need to limit the transparency of central banks policies are not borne out empirically – though a theoretical literature is willing to entertain the notion, as discussed further on. Theoretically, a greater and less volatile flow of information about its decisions should as likely be able to stabilize and rationalize rather than disrupt and corrupt financial markets. Indeed, it is probably true that the less accountable an agency, such as a central bank, the more transparent it should be. In other instances however, it is not evident that more information will strengthen financial systems. Examples to the contrary, wherein more information may worsen credit rationing or increase price volatility, are documented in [38]. Clearly more research, both conceptual and empirical, is needed to resolve this debate, its implications for the behavior and incentives of firms and individuals, and on economic outcomes.

The public good properties of information suggest a role for government in information acquisition and dissemination. Specifically this may entail creating rules and regulations specifying disclosure requirements about categories of information, frequency of disclosure, the standards for disclosed information and so forth. Moreover, since “perfect” information is rarely achieved even under the best of circumstances, there may be a need for government enforcement. Generally, transparency is limited by the inherent difficulties of obtaining information in rapidly changing environments. For example, sophisticated financial instruments which would make timely assessments of net worth of banks and firms are unreliable since markets respond sharply to constant changes in external factors. Achieving transparency may therefore not be sufficient. Enforcement mechanisms that ensure accountability by punishing fraudulent behavior, are also essential. In such circumstances, regulations may be necessary to minimize risks and ensure stability. As the following section suggests, such regulations need to balance the costs and benefits from increased disclosure in distinct circumstances.

Notably, transparency in and of itself is not sufficient without accompanying enforcement mechanisms. There is therefore a need for public institutions to both regulate disclosure and enforce appropriate behavior. Indeed, as illustrated by the case of Indonesia – financial reform may be predicated on broader public sector reforms. Notably, the effectiveness of public institutions affects not only the performance of markets C including capital markets – but also the allocation of public goods and the distribution of risk and other implicit costs in an economy. Subsuming more specific recommendations on transparency in financial markets, made earlier in this paper, are therefore broader imperatives to improve transparency in governance. Implementing freedom of information legislation, instituted freedom of press laws, built legal enforcement mechanisms, and invest in information infrastructure to improve the gathering and sharing of information. Many governments are already doing this.
Since institutions take to time to develop, successful medium term reforms often entail harnessing supplementary resources. These include soliciting private sector participation to elicit information about necessary levels of public participation, and promoting opportunities for exit in certain areas of service provision in order to give the citizens, as consumers, a chance to indicate their preferences.

Animating many of these reform initiatives is the principle of promoting transparency through public participation and “voice”. As studies increasingly demonstrate, “voice”, remains the fundamental means to ensure people’s right to information, to enforce transparency and accountability, and to thus improve the allocation of public resources. Empowering the voice of citizens with the necessary evidence through data and data analysis has proved to be successful in implementing transparency and effecting change.

Implementing voice entails addressing unique challenges. Though certain practices, such as decentralization, may encourage voice, they may weaken supervision and thus paradoxically reintroduce opportunities for corruption treated previously. More fundamentally, social and cultural circumstances may limit the ability of certain constituencies - the poor, women, children, from making their voice heard. Again, on this issue, as in the narrower problems treated above, more research is urgently needed.

4.1.4 Transparency with E-Government.

“E-Government has the potential to alter the traditional relationship between government and citizens by creating a new virtual government-and citizen interface” [79]. E-government, with ICT definition. There are common agreement that e-Government can enhance the transparency in traditional public administration [23, 14]. Although in different e-Government initiatives, different technologies be applied for achieving government performance evaluation, including accountability, participation, transparency and so on [23, 14, 36].

In paper [23], author applies a two stage multiple equation model and finds that internet use is positively associated with transparency satisfaction but negatively associated with interactivity satisfaction, and that both interactivity and transparency are positively associated with citizen trust in government. And transparency satisfaction in this work was operationalized as the level of reliability of the information provided by government websites.

And the Cyberspace Public Research Group 2001s, or CyPRGs, Web Attribute Evaluation System (WAES) provides two broad dimensions (interactivity and transparency) for evaluating federal websites that could quite easily be modified for evaluating websites of any level of government (or even the private sector). Additionally, in West’s research on state websites, assessing e-Government: the Internet democracy, and service delivery⁶. West outlines a methodology for evaluation that leads to substantive conclusions regarding the status and success of implementation: government websites are not making full use of available technology, and there are problems in terms of access and democratic outreach. Taken as a whole these studies are indicative of a general tendency in the literature to define criteria for evaluation and implementation in general, but do not give insight into the current status of implementation of specific functions [12].

However, narrative description of standardization and performance evaluation afterward are difficult to instruct e-Government system implementation as well as reduce risk

⁶http://www.insidepolitics.org/govtreport00.html
Transparency Types

The section of transparency of content may be usefully broken up into three parts:

- Information Transparency
- Operational Transparency
- Regulatory Transparency

According to the current understanding and researches of transparency in public administration and e-Government, we category transparency as the following three types, as *information transparency*, *operational transparency*, and *regulatory transparency*, and intent to combining the transparency with the relative factors which was mentioned in above subsection, as well as, make the transparency idea clear.

**Information Transparency** From the literature study, we understand information transparency, as, Information reliability and the accessability/openess of government information.

**Operational Transparency** By operational transparency, we understand it as how much the citizen be involved into the government procedure. So that, the degree of citizen participation in government procedure should be considered as an important factor of operational transparency. At the same time, the more opacity of government operation, the more possibility to predict the result of the government, therefore, the predictability is also close connected with operational transparency.

**Regulatory Transparency** Regulatory transparency, refers to a product of the accessibility and intelligibility of laws, on the one hand, and the openness and consistency of the processes by which they are made, on the other. That is, it involves both the law-making process and the implementation and enforcement of the law [63].

Related Factors and Concepts

As the diversity understandings of the transparency; the transparency studies in public governance are also combined with many other critical factors and concepts which closely related to “good governance”. So we go though these related factors and concepts, for illustrating transparency more deeply and extensively.

a) **Corruption**: in common usage the word ”corruption” is used to mean different things in different context [59]. It can involve the misuse of policy instruments, illicit or licit services and so on [51]. It usually be denoted as the opposite side of transparency, as well as, can be reduced by enhancing transparency and accountability in public government [73, 33].

b) **Accountability**: as more or less synonymous as ‘answerable’, usually understood as the providing of answers, as reporting or, more obviously “giving an account” [36]. Some researchers on central bank also believe that an important practical prerequisite for accountability is policy transparency, moreover, transparency facilitates the task of monitoring the government agency (eg. Central Bank) and of holding it accountable [1].
c) **Openness** and **Accessibility**: both of these two terms are often used for describing the degree of accessing government information from web service by the majority of the citizens, in e-Government researches [31, 8, 24, 61]. It is a more countable factors and W3C’s WCAG was also used for the purposed of its evaluation. Information accessibility usually be considered as an important factor of transparency [73, 63, 31, 8].

d) **Predictability**, this terminology shows most frequency in the transparency problem of government monetary policy [26, 61, 60]. Some researchers use it as a factor to measure transparency [26]. Generally speaking, the more opacity of the system, the accurate results of running this system can be predicted [26].

e) **Participation**, is considered as a criterion of transparency [27]. It is generally accepted that the participation process should be transparent, so that the wider public can see what is going on and how decisions are being made [52, 27].

f) **Reliability**, especially information reliability, as a general term not only used in the feeling of citizen to government, but also information disclosure of other systems [23, 25]. It is a kind of individual satisfaction, therefore, is also used as an appropriate measure of the transparency construct, we can conclude if government, according to citizens, is doing a good job in the transparency area [23].

Therefore, transparency can be demonstrated as a interaction of other factors:

\[
\text{Transparency} = \left(\text{Accountability} + \text{Accessibility/Openness} \right) + \text{Participation} + \text{Reliability} + \text{Predictability} + \text{Anticorruption}
\]

Furthermore, “citizen trust” and “democracy” are very important concepts, and often mentioned by public administration and e-Government researches. They are often viewed as a result for transparency and other relative factors [23, 14, 13, 10].
The Degree of Government Transparency

From above study, transparency is a desired property in government systems as well as also no government dare to say its completely transparent to their citizen, because of many issues, such as private issues, security issues and so on. Therefore, for easy evaluation and judging, we follow the method of sensitive information classification, and divided transparency into different levels. The figure 4.2 illustrates the transparency degree, where the basic transparency starts from level 0, while the government system meets the descriptions of level 0 it has the weakly transparency, and while the government system fulfills the description of transparency level 3, it enjoys the most strongly transparency.

As the illustration in figure 4.3, transparency level 0 is defined as: the system has basic operational transparency, namely no improper operations happened and information transparency, namely basic document tracing and policy information access is available. Transparency level 1 is described as: operational transparency and information transparency is same as level 0, and the citizen can join, at least trace the rule, regulation, policy and law document making. Transparency level 2 is presented as: all document information is getable but detailed operation information may be not. Transparency level 3 represent completely transparency, which refers to citizen can access every information, know every actions of government and involved in every rule, regulation and law generation.

Depending on different government cases, the systems can achieve different level of transparency can be understand of transparent. For instant, in social welfare case, the citizen can pursue as strongly transparency of the system as possible, therefor in this case, the system which achieve as least level 2, we consider it as transparent; however, in government financial case, some sensitive information may be included, if the citizen in
the system can get transparency level 1, we consider this system is transparent enough.

4.2 Transparency Evaluation Methodology

Depending on different types of transparency, various measuring methods have been proposed by researchers in public administration and e-Government fields [26].

The one of the most popular method used for measuring transparency is that give survey on the citizen satisfaction on public service, information accessability, citizen trust and so on [23, 13]. In monetary policy area, economic equations also be imported for analysis transparency and factors around this concept [35, 7]. As the criterion of transparency, some researchers even proposed frameworks for evaluation participation [27].

In paper [23], author applies a two stage multiple equation model and finds that internet use is positively associated with transparency satisfaction but negatively associated with interactivity satisfaction, and that both interactivity and transparency are positively associated with citizen trust in government. And transparency satisfaction in this work was viewed as the level of reliability of the information provided by government websites.

And the Cyberspace Public Research Group 2001’s, or CyPRG’s, Web Attribute Evaluation System (WAES) provides two broad dimensions (interactivity and transparency) for evaluating federal websites that could quite easily be modified for evaluating websites of any level of government (or even the private sector). Additionally, in his research on state websites. Assessing e-government: the Internet democracy, and service delivery7. West outlines a methodology for evaluation that leads to substantive conclusions regarding the status and success of implementation: government websites are not making full use of available technology, and there are problems in terms of access and democratic outreach. Taken as a whole these studies are indicative of a general tendency in the literature to define criteria for evaluation and implementation in general, but do not give insight into the current status of implementation of specific functions [12].

However, narrative description and performance evaluation afterward are difficult to instruct e-Government system implementation as well as reduce risk of e-Government project.

4.3 Formal Understanding of Transparency

4.3.1 Formal Description of Transparency based on GSD Model

A formal technique is a notation with a mathematical semantics: a precise explanation of how it should be interpreted, given in terms of an already-understood area of mathematics - such as algebra, logic, or set theory. There are already some formal method applications in e-Government initiatives, but most of these applications are focus on technologies but neglect of the nature of public management and governance. To this end, we believe that our work on applying formal technique into discussion of the nature of public management or governance, is a new experimentation and can work for reducing the high rate of failures in e-Government project [34].

7http://www.insidepolitics.org/govtreport00.html
4.3.2 Formal Description of Transparency based on GSD Model

The word “transparency” carries with it a powerful array of moral and political associations, including honesty, guilelessness, and openness [32]. Since the concept of transparency itself is controversial, we give some hypothesis of the model based on our emphasis on analysis quality of e-service over e-government, before we enter the formal discussion part of transparency.

**Hypothesis 1**: The ability and condition of citizen who access e-service from government are same. We focus on analyzing how to ensure transparency property in government workflow itself by formal method, therefore, we ignore the distinguish of subjective condition of citizens.

**Hypothesis 2**: The policies of government are consistency while be applying on e-government service.

By the definition of transparency, we take your suggestion, to look into transparency also its opposite side, namely, corruption. So we re-define the transparency as follows:

A) Look at the opposite side of transparency C corruption (if the following actions happened, the officer may get chance to be corrupt).

a. An officer cannot evaluate his/her own application forms.

b. An officer should not be allowed to issue his/her own benefit.

c. An officer cannot perform all the tasks in a same workflow instance.

B) Transparency: the quality of something, that allows you to see through it.

a. Applicant can track his/her application.

b. Information about the policy is known before execute.

c. Decision made with reason.

Corresponding with the narrative descriptions, the formal descriptions of transparency based on GSD model can be expressed as follows:

\[
\text{eq inv1}(S,A,D) = \\
[A,\text{evaluate}] /\text{in} \text{ docTrace}(\text{doc}(S,D)) \text{ or } [A,\text{decide}] /\text{in} \text{ docTrace}(\text{doc}(S,D)) \\
\implies \text{not}([A,\text{submit}] /\text{in} \text{ docTrace}(\text{doc}(S,D))) .
\]

\[
\text{eq inv2}(S,A,D) = \\
[A,\text{execute}] /\text{in} \text{ docTrace}(\text{doc}(S,D)) \\
\implies \text{not}([A,\text{submit}] /\text{in} \text{ docTrace}(\text{doc}(S,D))) .
\]

\[
\text{eq inv3}(S,A1,A2,A,D) = \\
[A1,\text{evaluate}] /\text{in} \text{ docTrace}(\text{doc}(S,D)) \text{ and } [A2,\text{execute}] /\text{in} \text{ docTrace}(\text{doc}(S,D)) \\
\implies \text{not}(A1 = A2) .
\]

-- these wrote by direct definition of transparency -- actor can track

\[
\text{eq inv4}(S,A,D) = \\
c-\text{track}(S,A,D) \text{ and not(D }/\text{in} \text{ basicLawSet}) \implies [A,\text{submitAD}] /\text{in} \text{ docTrace}(\text{doc}(S,D)) .
\]

-- information about policy know before execute

\[
\text{eq inv5}(S,A) = \\
\text{subset}((\text{basicLawSet},\text{actorKnow}(\text{actor}(S,A))) .
\]

-- decision made with reason.

\[
\text{eq inv6}(S,A,D) = \\
[A,\text{submitAD}] /\text{in} \text{ docTrace}(\text{doc}(S,D)) \text{ and } \text{docType}(\text{doc}(S,D)) = \text{ddoc} \\
\implies \text{subset}(\text{docRefer}(\text{doc}(S,D)),\text{actorKnow}(\text{actor}(S,A))) .
\]

-- accountability

\[
\text{eq inv7}(S,A,D) = \\
\text{docLabel}(\text{doc}(S,D)) = \text{verified} \implies [A,\text{evaluate}] /\text{in} \text{ docTrace}(\text{doc}(S,D)) .
\]

-- information integrety

\[
\text{eq inv8}(S,A,D) = \\
([A,\text{submitAD}] /\text{in} \text{ docTrace}(\text{doc}(S,D)))
\]
implies (docInfo(doc(S,D)) < sumInfo(actorKnow(actor(S,A)))) .

4.3.3 Formal Discussion of Transparency

We describe the verification of the desired property \texttt{inv1} of the GSD model. Following the method described in Chapter 2, we first declare a module named as \texttt{ISTEP}, which imports the module \texttt{INV1}, to describe the predicate to be proved in each inductive case. The following operator and equation are declared, where \( s \) and \( s' \) are constants of sort \texttt{Sys} declared in the module \texttt{ISTEP}:

\begin{verbatim}
op istep1 : ActorName DocName -> Bool
eq istep1(A,D) = inv1(s,A,D) implies inv1(s',A,D) .
\end{verbatim}

We do structural induction to verify the property. The property denoted by \texttt{inv} reduces to \texttt{true} on any initial state. And in the inductive cases, we have two subcases in which \texttt{ISTEP}. The corresponding three proof passages are as follows:

--- 1) apply(s)
-- 1.1) c-apply(s,a,d,i)
  open ISTEP
  op i : -> Info .
-- arbitrary objects
-- assumptions
  eq [a,apply] /in docTrace(doc(s,d)) = true .
  eq c-apply(s,a,d,i) = true .
  eq s' = apply(s,a,d,i) .
  red istep1(a1,d1) .
  close

open ISTEP
op i : -> Info .
-- arbitrary objects
-- assumptions
  eq [a,apply] /in docTrace(doc(s,d)) = false .
  eq c-apply(s,a,d,i) = true .
  eq s' = apply(s,a,d,i) .
  red istep1(a1,d1) .
  close

-- 1.2) not c-apply(s)
-- successor state
  open ISTEP
  op i : -> Info .
  eq c-apply(s,a,d,i) = false .
  eq s' = apply(s,a,d,i) .
  red istep1(a1,d1) .
  close

Even if the three proof passages returns false for the desired property, we can use lemmas to strengthen the inductive hypotheses in the proof passages by reducing. So we can prove property which described in \texttt{inv1} is an invariant based on GSD model. Therefore, the GSD model fulfills this aspect of basic transparency descriptions. The other formal descriptions of transparency can be discussed by CafeOBJ introductive prover in the same way.
4.4 Summary

In an area as complex and controversial as transparency, no single source or polling method has yet been developed that combines a perfect sampling frame, large enough country coverage, and a fully convincing methodology to produce comparative assessments. This is why we look this concept from different angles and classify it by different standards. To test our understanding and the method we proposed, as chemical experiment, we need apply it into different cases and different situation. Therefore in the following chapter, we try our method into three different cases in government systems, which are:

- Social Welfare (C2G)
- Business License (B2G)
- Financial Budget Disclosure (G2G)
Chapter 5

Three Case Studies for Transparency Discussion

*A theory is something nobody believes, except the person who made it. An experiment is something everybody believes, except the person who made it.*

Albert Einstein

In this chapter, we work as a chemist rather than a computer scientist. We designed a methodology in Chapter 3 and set experiment target (transparency degree) in Chapter 4. As a chemist has prepared test tube, experiment environment and test material, we are starting to test the systems’ transparency under the environment providing by formal technology (CafeOBJ).

This chapter is organized by three case studies to check the usability and feasibility of the property discussion method we proposed in previous chapters. The three cases try to cover as many branches of e-Government implementations as we can image. Therefore, the transparency discussions can include the actor who is inside the systems as well as the one who is outside the systems.

- Social Welfare (C2G; citizen is exactly inside the service delivery system)
- Business License (B2G; citizen who are not involved in business parties, are not the stake-holders of the system)
- Financial Budget Disclosure (G2G; only government officials involved in the process of budget making, however, the budget is related with every citizen)

Our method is not panacea for solving government transparency problems for sure, however, we are providing a constructive suggestion from views of software engineering and formal method for supporting to realize a more transparent e-Government system.

5.1 Social Welfare

5.1.1 Narrative Description of Social Welfare

Welfare is “the state of being or doing well: the condition of health, prosperity, and happiness; well-being.” A benefit is “anything that is good for a person.” A social
welfare benefit may have a material, therapeutic, or opportunity objective [30]. Benefits may be combined to accomplish multiple objectives. For instance, the Job Corps program provided financial aid, counseling, and vocational training in one package.

Social welfare policy announcement and benefits delivery may be various. For example, benefits may be delivered as in-kind goods and services, cash, or vouchers. Which method is best? What kind of service can be considered more transparency to applicant (citizen)? Let us start these discussion from the social welfare service delivery work-flow, which be shown in the figure.

The modeling method of the social welfare case is conducted on the GSD model which is mentioned in chapter3. The social welfare service is considered as a sample workflow with related security and transparent considerations, which is used for delivery social benefit.

The figure shows the processing procedure of social welfare service delivery. The workflow consists of five tasks; and the states of the workflow changed after execution of each task. Initially, the state of the workflow is start. In the first task apply, an citizen applies for social benefit by filling out an application form. Two local administrators have to evaluate and give decision on this application (tasks verify and decide). Based on the evaluation results of the administrators, the decision will be reported (task report) if both
administrators approved this application; and official will issues the benefit or the official will notify the citizen that his/her application has been rejected (task issues) if either of the managers rejected it.

We considered the basic constrains will developing this kind of systems, such as, RBAC mechanism, static SoD (SSoD) and dynamic SoD (DSoD) constraints involved in the workflow; and exam in this kind of environment the transparent degree this system will be. There are three roles involved in this workflow: processing officer, collaborative agency and citizen. All users belong to role citizen; each user that belongs to role processing officer or collaborative agency also belongs to role citizen, but a user who belongs to role processing officer, cannot belong to role collaborative agency, and vice versa (i.e., the roles processing officer and collaborative agency have no shared users). SSoD constraints demand that different tasks are executed by different roles [?]. In other words, task apply can be executed by users belonging to role citizen (i.e., all users), tasks verify and decide can only be executed by users belonging to role processing officer. DSoD constraints demand that a user belonging to multiple roles (such as, a user belonging to roles citizen and processing officer) can dynamically activate only one of these roles (citizen or processing officer) in the run-time.

5.1.2 The OTS/CafeOBJ Specification of the Social Welfare Case

The OTS model of the social welfare model is defined in a CafeOBJ module with the name PA\_SW using the CafeOBJ keyword mod*, which indicates that the module is a loose semantics declaration, meaning an arbitrary model (implementation) that respects all requirements written in the module. The module PA\_SW imports all the data type modules defined in advance. A hidden sort Sys is declared in the module as *[Sys]*, by enclosing it with *[ and ]*, which denotes the universal state space of the OTS model.

In module PA\_SW, two observers denoted by CafeOBJ observation operators actor and doc are declared as follows:

\[
\begin{align*}
\text{bop actor} & : \text{Sys ActorName} \rightarrow \text{Actor} \\
\text{bop doc} & : \text{Sys DocName} \rightarrow \text{Doc}
\end{align*}
\]

We consider all citizen, and processing as actor in this system. And given a state (say s) of the OTS and an actor’s name (say an) and a document name (say dn), observers actor, returns the status (value of each component) of the actor with the name an and being working on the document dn in the state s. And given a state of the OTS, observer doc returns the status (value of each component) of the document with the name dn.

A constant init is declared as “op init : -> Sys” to denote any initial state of the OTS model of the PA system. The initial state is characterized by the following equations:

\[
\begin{align*}
\text{-- [0] equations of initial state} \\
\text{eq actor(init,C)} &= \text{mk-actor(C, idling, basicLawSet, issued).} \\
\text{eq doc(init,D)} &= (\text{if D /in basicLawSet then criteria else emptyDoc fi }) \\
\end{align*}
\]

In the initial state of the social welfare system, we suppose that there are some basic social welfare policy documents which have been published, and both citizen and officers should start the public administration procedure based on these basic law document set.
In the first equation, variable \( C \) denotes an arbitrary actor’s name (note that the name component is the identifiers of both actors and documents). The right-hand side of the equation describes the components of actor with the name \( C \). The actor components are composed using the operator \( \text{mk-actor} \). Any actor denoted by \( C \) is initially in the status \( \text{idling} \), and have knowledge with some basic law \( \text{basicLawSet} \). The second equation says that at the initial stage of this system, there are some basic law documents, if a document named \( D \) are not in basic law document set then this document should be not existed \( \text{basicLawSet} \).

Based on the narrative description in previous section, six transitions, which characterize the processing of documents and track document, are declared as follows:

\[
\begin{align*}
\text{-- actions} \\
\text{bop submit} & : \text{Sys ActorName DocName InformationSet} \rightarrow \text{Sys} \\
\text{bop verify} & : \text{Sys ActorName DocName DocNameSet Info} \rightarrow \text{Sys} \\
\text{bop decide} & : \text{Sys ActorName DocName Info} \rightarrow \text{Sys} \\
\text{bop report} & : \text{Sys ActorName DocName Info} \rightarrow \text{Sys} \\
\text{bop issues} & : \text{Sys ActorName DocName Info} \rightarrow \text{Sys} \\
\text{bop track} & : \text{Sys ActorName DocName} \rightarrow \text{Sys}
\end{align*}
\]

Take transition denoted by the CafeOBJ action operator \( \text{verify} \) as an example. It characterizes that actor who belongs to processing officer role \( \text{processing officer} \) verify an application document created by a citizen, and transfers the evaluated application document to another officer. The effective condition of the transition is as follows:

\[
\begin{align*}
\text{-- [2] verify} \\
\text{op c-verify} : \text{Sys ActorName DocName DocNameSet Info} \rightarrow \text{Bool} \\
\text{eq c-verify}(S,C,D,DS,I) \\
& = C \in \text{agenciesSet} \text{ and not } (C = \text{docCreator(doc(S,D))) and}
\text{docType(doc(S,D))) = \text{pdoc and docLabel(doc(S,D))) = applied}}
\text{and (auth(docCreator(doc(S,D)))\text{,verify\text{,agenciesSet) /in docAuth(doc(S,D)))}}.
\end{align*}
\]

The condition denoted by \( \text{c-verify} \) says: there is an application document \( D \) in the status \( \text{applied} \); the actor \( C \) who is going to verify the application document is an officer of public agency \( \text{agenciesSet} \); actor \( C \) is not the creator of document \( D \); and his/her role has right to perform this verification.

The CafeOBJ specification describing execution of transition \( \text{verify} \) is shown above as well. If \( \text{verify} \) is applied when the condition holds: (1) The state of actor who
apply this action verify change into handling and the knowledge and track about his own document of this actor do not change. (2) The document which be evaluated, will record the evaluation information docInfo, who evaluated it in document trace, delete the evaluation right for this role and authorize other role for next processing step. (3) If verify is applied when the effective condition does not hold, nothing changes.

The remaining transitions can be defined in a similar way, which are omitted here.

5.1.3 Transparency Discussion of Social Welfare Case

So which level of transparent property this model will fulfill and is this model transparent enough to citizen who make use of it? Is there any improve possibility for this model be more transparent?

- A officer should not be allowed to evaluate his/her own application form.
- A officer should not be allowed to issues his/her own social benefit.
- A officer should not be allowed to perform both verify and decide tasks in a same workflow instance.

For above basic transparent requirement, we can find they have some inter-relationship with the basic transparency descriptions in previous chapter. Therefore, we are going to introduce simulation proof to discuss it. The details information about simulation proof can be found in section 5.4. By proving simulation relation between case and GSD model, we do not need verify the properties again, which have been held in GSD model, in particular cases again.

And in this concrete case, by adding more information descriptions in the module INFORMATION, we can discuss more properties, which achieve more high level transparency.

- Two citizens who are in same situations, are treated as same way.
- Every actions make with their application are reported to the applicant.
- Application context only can be edited by applicant.

For the last issue, since the officer’s “verify” and “decide” actions can change application as well. To prove it, we need more detailed division on information, e.g. divide into application context and officer opinion. Because the social welfare law is closely related with citizen interest, if citizen can involve into relative law making procedures, the system can fulfill a kind of high level transparency. However, in current government service delivery model, the law making procedure has not been considered.

5.2 Business Licensing

5.2.1 Narrative Description of Business Licensing

In order to uphold public health and safety or preserve the citizen or other stake-holder’s rights, business people are expected to comply with licensing requirements and operate
only once a license has been granted. Business people who do not comply with licensing
requirements bear individual responsibility and are liable to punishment. Engaging many
people’s interest and benefit during the granting licenses procedures, therefor in the de-
veloping world business licensing practices have been described as, “so corrupted they are
simply income opportunities for those charged with enforcing regulation” [57]. So it is
necessary to discuss the transparency property around this case.

For business licensing service refers all the relevant Government registrations and
licences for conducting business. In this service, it usually involves more government
agency message exchanges and supplementary information. So the model of business
licensing is larger than the previous example and the analysis part is contains more steps
as well.

a Applying for new license (this step contains almost as same sub-steps as in previous
social welfare case);

b Renewing of licenses;

c Re-instating revoked licenses;

d Tracking the status of license requests;

e Booking pre-application meetings.

By applying for new license, the process is basically same with social welfare applica-
tion except with some more steps with public agencies collaboration. The applying for new
license are including the following steps: (a) Submission of license application; (b) Sub-
mission of supporting documents Verification of applications and supporting documents;
(c) Eligibility assessment Requesting more information from applicants, if required; (d)
Internal assessment of applications; (e) Decision making on applications based on inter-
nal assessment and external inputs; (f)Notification of applicants; (g) Issuance of licenses.
And the process of renewing licenses and re-instating revoked licenses are basically same
with the new license application, except the application documents are different. And
by tracking the status of license requests and booking pre-application meetings, there are
two transition actions to describe them.

The figure shows the process of business license service delivery. The new license
applying workflow consists of eight tasks; and the states of the workflow changed after
execution of each task. Initially, the state of the workflow is start. In the first task
submit_application, a business party applies for business licenses service by filling out an
application form. submit_supplementray, and then submit corresponding supplementary
documents. Corresponding public sectors have to verify the application (task verify), and
contact (task contact) collaborating public agency for information. Based on the collabo-
rative public agency’s feedback(task feedback), the processing office will give decision(task
decide). At last, the decision will be notified (task notify) to all stake-holders. And au-
thority sector will issues the business license to applicant(task issue) if the application
has been approved.

5.2.2 The OTS/CafeOBJ Specifications of the Workflow

The OTS model of the business license model is defined in a CafeOBJ module with the
name PA\\_BL using the CafeOBJ keyword mod*, which indicates that the module is a
loose semantics declaration, meaning an arbitrary model (implementation) that respects all requirements written in the module. The module PA\_BL imports all the data type modules defined in advance. A hidden sort Sys is declared in the module as *[Sys]∗ by enclosing it with *[ and ]∗, which denotes the universal state space of the OTS model.

The basic modeling method is same as we present in the previous section, only this workflow is involved in more actor roles, such as: approving authority, collaborating agencies and etc., and has more transition actions. Therefore, in the business license service system, the workflow is considered as a composition of two parts: (1) a workflow process, which consists of a collection of tasks organized in a logic order, and (2) booking pre-application meeting and application tracing, which are associated with the workflow process. By the increasing of actor roles, we revise the ACTORROLESET sort of the GDS model by adding actor roles as follows:

```plaintext
mod! ACTORROLESET
  pr (SET(P <= view to NAME sort Elt -> ActorName)
      *sort Set -> ActorRoleSet)
--
-- three roles of the public administration system
ops applicantSet collaboratingSet approvingauthoritySet agenciesSet : -> ActorRoleSet
--
op __=__ : ActorRoleSet ActorRoleSet -> Bool comm
var ARS : ActorRoleSet
eq (ARS = ARS) = true .
...```

Figure 5.2: Illustration of Business License Service Workflow
We consider all citizen, processing officer and collaborative agency as actor in this system. And given a state (say $s$) of the OTS and an actor's name (say $an$) and a document name (say $dn$), observers $actor$, returns the status (value of each component) of the actor with the name $an$ and being working on the document $dn$ in the state $s$. And given a state of the OTS, observer $doc$ returns the status (value of each component) of the document with the name $dn$.

\[
bop\actor : \text{Sys }\text{ActorName} \rightarrow \text{Actor} \\
bop\doc : \text{Sys }\text{DocName} \rightarrow \text{Doc}
\]

There are nine transitions in the OTS according to tasks in the narrative description. In the following, we will describe the effective conditions of these transitions, and how these transitions change the states of the OTS. Note that here a state of the OTS, in a broad sense, includes: states of the renewing license, booking pre-application meetings, and states of tracing the status of application (i.e. elements in authorization and execution histories). The OTS for the workflow is written in CafeOBJ. The complete CafeOBJ specifications of the workflow can be found in Appendix C, and here we only show some representative parts of the OTS module. The hidden sort denoting the states of the OTS model for the workflow is declared as $\text{Sys}$. The operators denoting the observers and transitions are declared as follows:

\[
\begin{align*}
\text{-- observations -- initial state} \\
& \text{op init : } \rightarrow \text{Sys} \\
\text{-- observer} \\
& \text{bop actor : Sys ActorName }\rightarrow\text{Actor} \\
& \text{bop doc : Sys DocName }\rightarrow\text{Doc} \\
\text{-- actions} \\
& \text{bop submit : Sys ActorName DocName DocNameSet Info }\rightarrow\text{Sys} \\
& \text{bop verify : Sys ActorName DocName DocNameSet Info }\rightarrow\text{Sys} \\
& \text{bop contact : Sys ActorName DocName Info }\rightarrow\text{Sys} \\
& \text{bop feedback : Sys ActorName DocName Info }\rightarrow\text{Sys} \\
& \text{bop decide : Sys ActorName DocName Info }\rightarrow\text{Sys} \\
& \text{bop notify : Sys ActorName DocName Info }\rightarrow\text{Sys} \\
& \text{bop issue : Sys ActorName DocName Info }\rightarrow\text{Sys} \\
& \text{bop book : Sys ActorName DocName Info }\rightarrow\text{Sys} \\
& \text{bop track : Sys ActorName DocName }\rightarrow\text{Sys}
\end{align*}
\]

$\text{ActorName}$, $\text{DocName}$, $\text{DocNameSet}$, and $\text{Info}$ are the sorts denoting: applicant of this workflow, application forms to be handled by the workflow, supplementary documents added with the application and information involved in this transition action. The corresponding data type modules defining these sorts are imported in the OTS module.

The initial state of the OTS is defined with the following equations:

\[
\begin{align*}
\text{-- [0] equations of initial state} \\
& \text{eq actor}(\text{init},C) = \text{mk-actor}(C,\text{idling},\text{basicLawSet},\text{issused}) \; . \\
& \text{eq doc}(\text{init},D) = (\text{if } D /\text{in} \text{ basicLawSet then criteria else emptyDoc } \text{fi}) \; .
\end{align*}
\]

Here we take the transition action $\text{decide}$ as an example to illustrate the transition action description as follows. The effective condition of transition $\text{decide}$ is denoted by $\text{c-decide}$, which includes: (1) the demands for the process conditions: actor should be in
public agency role agenciesSet, and the actor C is not the creator of application document, and (2) the demands for the authorization conditions: role agenciesSet should have been granted the privilege to decide task decide on the current application form D, and the document label shows (say, verified) this application is proceed on stage verified and this document is still private document (pdoc) can only be tracked by applicant.

```
-- [3] decide
op c-decide : Sys1 ActorName DocName Info -> Bool
eq c-decide(S,C,D,I)
  = C /in agenciesSet and not(C = docCreator(doc(S,D)))
      and not(docVerifier(doc(S,D)))
      and not(C = left(getTerm(verify,docTrace(doc(S,D)))))
      and docType(doc(S,D)) = pdoc and docLabel(doc(S,D)) = verified and
      (auth(left(getTerm(verify,docTrace(doc(S,D))))) , decide, agenciesSet)
      /in docAuth(doc(S,D)))

--
ceq actor(decide(S,C,D,I),C1)
  = (if not(C1 = C) then actor(S,C1) 
    else mk-actor(C, handling, actorKnow(actor(S,C)), actorTrack(actor(S,C))) fi)
if c-decide(S,C,D,I)

cseq doc(decide(S,C,D,I),D1)
  = (if not(D1 = D) then doc(S,D1) 
    else mk-doc(D, docCreator(doc(S,D)), docStatus(doc(S,D)), docType(doc(S,D)),
       (I @ docInfo(doc(S,D))), add([C, decide], docTrace(doc(S,D)))))
    del(auth(left(getTerm(verify,docTrace(doc(S,D))))) , decide, agenciesSet),
    add(auth(C, report, agenciesSet), docAuth(doc(S,D))))
    (actorKnow(actor(S,C)) U docRefer(doc(S,D))), decided) fi)
if c-decide(S,C,D,I)

cseq decide(S,C,D,I) = S
if not c-decide(S,C,D,I)

And the other transition actions are described by the same way.

5.2.3 Transparency Discussion of Business Licensing Case

In business licensing case, the basic transparency can be discussed in the same way as the way we have illustrated in social welfare case. And more high level transparency property can be discussed by giving more detailed description on supporting document and actor’s information on rule and regulations. For instance, we can describe the following properties around high level transparency:

- Applicant’s supporting document are staying same after submitted applications.

- Related rule and regulation about business license keeps same during an application period.

- A citizen, who does not involve in the business license proceed, can trace the license document after it has been issued.

As social welfare case, if citizen can involve in to relative law making procedures, the system can fulfill a kind of high level transparency. However, in current service delivery model, this part has not been described in.
5.3 Financial Budget Disclosure

5.3.1 Narrative Description of Fiscal Disclosure

Fiscal disclosure in government can be seen as a indicator of the government’s performance or capacity, in raising, handling, and using public money. With such information, one potential benefit of government budget and financial reports are to help people make better decisions about their community, their government, and their economy. These decisions may relate to the election of officials, votes on new projects, and even the decision to stay in or move away from a community. Further, the reporting may provide information so that decisions that make the jurisdiction better off. So transparency of public financial information and budget making are often considered one of the cornerstones of good democratic governments. Financial reports and budget are based on accounting rules and other types of rules or standards that often capture the day-to-day monetary transactions and events of government. The transactions are then summarized into financial reports or reflected on budget making. As we discussed the category of transparency in e-Government systems, the transparency of budget making and public financial reporting covers all above three types. In this case, we are mainly focus on government budget making process.

Table 5.1 is a brief overview on government annual budget cycle. Two centuries of evolution have produced a complex budget process, with many players and stages, defined rules and roles, and layers of procedure [2]. Each reform has deposited its distinctive requirements; some have faded away, and others have been incorporated into the ongoing activities of budgeting. The various requirements revolve around an annual cycle that begins with the formulation of the president’s budget in the executive branch, involves four separate sets of congressional action, then moves to agencies, which implement their approved budgets, and concludes with the review and audit of expenditures. Table lists the major actions taken during the fiscal year 2000 budget cycle. While the time table is constructed on the assumption that each action is completed on schedule, it has become common for congress and the president to miss certain deadlines. Nevertheless, the basic steps are repeated year after year with little change, though particular procedures may vary with the style of the president, his relations with congress, or in response to changes in economic or political conditions. Table indicates that the federal government has an annual budget process that sprawls over several years. This extended cycle means that at any given date, the federal government is typically juggling three fiscal years. The overlap of several fiscal years has made budgeting an arduous, year-round activity, which allows little time for pause and reflection and little opportunity for thinking strategically about the purposes and direction of federal programs. In budgeting, the calendar is king, telling participants what they are supposed to do and when. Even when deadlines are missed, the calendar shapes actions and expectations, and it patterns the roles and behavior of those who make budgets and the spend money. The most important date on the budget calendar is October 1, the day the new fiscal year begins (this day is different from country to country). Agencies begin compiling their budgets with this date in mind, the president sends his budget to Congress mindful of the time required for legislative action, Congress organizes much of its work around this date, and agencies open and close their books on the basis of the October 1 starting point.
<table>
<thead>
<tr>
<th>Calendar</th>
<th>Activities</th>
</tr>
</thead>
<tbody>
<tr>
<td>1998 March- June</td>
<td>Budget guidelines and preliminary policies are developed; agency budget office issue call for estimates to operating units.</td>
</tr>
<tr>
<td>July - September</td>
<td>Agencies formulate detailed requests, which are submitted to OMB.</td>
</tr>
<tr>
<td>October - December</td>
<td>OMB reviews agencies’ requests and issues passbacks; agencies appeal to OMB and or the president final decisions are made.</td>
</tr>
<tr>
<td>1999 January</td>
<td>CBO releases Economic and Budget Outlook for fiscal 2000</td>
</tr>
<tr>
<td>February</td>
<td>President submits his fiscal 2000 budget to Congress no later than the first Monday in February.</td>
</tr>
<tr>
<td>March 15</td>
<td>Congressional committees submit views and estimates on the budget to Budget committees.</td>
</tr>
<tr>
<td>April 15</td>
<td>Congress adopts the fiscal 2000 budget resolution.</td>
</tr>
<tr>
<td>May 15</td>
<td>If budget resolution had not been adopted, appropriations could be considered in the House.</td>
</tr>
<tr>
<td>June - August</td>
<td>House and Senate act on regular appropriation bills for fiscal 2000; OMB and CBO release new revenue and expenditure projections for fiscal 2000.</td>
</tr>
<tr>
<td>September</td>
<td>Conference reports and enactment of regular appropriations.</td>
</tr>
<tr>
<td>October 1</td>
<td>Fiscal 2000 starts; continuing resolution is passed since regular appropriations have not been enacted.</td>
</tr>
<tr>
<td>October - November</td>
<td>CBO and OMB issue PAYGO reports for the past session of Congress.</td>
</tr>
<tr>
<td>October 1999- September 2000</td>
<td>Agencies spend resources and carry out activities and authorized by Congress; Congress may make supplemental appropriations for fiscal 2000.</td>
</tr>
<tr>
<td>2001 January - December</td>
<td>Agencies prepare financial statements; postaudits and evaluation are conducted.</td>
</tr>
<tr>
<td>February</td>
<td>Actual revenue and expenditure data for fiscal 2000 are included in the fiscal 2002 budget.</td>
</tr>
</tbody>
</table>
5.3.2 The OTS/CafeOBJ Specifications of the Workflow

By the

There are eight transitions in the OTS according to tasks in the narrative description. In the following, we will describe the effective conditions of these transitions, and how these transitions change the states of the OTS. Note that here a state of the OTS, in a broad sense, includes: states of the renewing license, booking pre-application meetings, and states of tracing the status of application (i.e. elements in authorization and execution histories).

The basic modeling method is same as we present in the previous section, only this workflow is involved in more actor roles, such as: approving authority and etc., and has more transition actions. Therefor, in the business license service system, the workflow is considered as a composition of two parts: (1) a workflow process, which consists of a collection of tasks organized in a logic order, and (2) booking pre-application meeting and application tracing, which are associated with the workflow process. By the increasing of actor roles, we revise the ACTORROLESET sort of the GDS model by adding actor roles as follows:

-- actions
bop guide : Sys ActorName DocName InformationSet -> Sys
bop request : Sys ActorName DocName DocNameSet Info -> Sys
bop evaluate : Sys ActorName DocName Info -> Sys
bop feedback : Sys ActorName DocName Info -> Sys
bop appeal : Sys ActorName DocName Info -> Sys
bop decide : Sys ActorName DocName Info -> Sys
bop release : Sys ActorName DocName -> Sys
bop submit : Sys ActorName DocName InformationSet -> Sys
bop discuss : Sys ActorName DocName DocNameSet Info -> Sys
bop adopt : Sys ActorName DocName Info -> Sys
bop report : Sys ActorName DocName Info -> Sys
bop enact : Sys ActorName DocName Info -> Sys

And by the increment of business procedures, the OTS model involved more transition actions for describing them. According to the adding processes, we keep two observers in the OTS: actor and document. Among them, the first observer observes all three current actor roles’ states; and the last observer observes current authorizations defined on the three sub-processes with the purpose to check: currently, privilege to execute some task has been granted to which role. And such a privilege is denoted by an authorization. For each workflow instance, there exist one authorization history of actual authorizations granted to roles and one execution history of actual executions of tasks by subjects. Authorization/Execution history is denoted by a queue, whose elements are authorization/execution records. During the execution of workflow, all authorizations that have been granted to roles (i.e. authorization records) are added to the the queue of authorization history, and all executions of tasks by subjects (i.e. execution records) are added to the queue of execution history.

And by the increment of business procedures, the OTS model involved more transition actions for describing them. According to the adding processes, we keep two observers in the OTS: actor and document. Among them, the first three observers observe current states of the three sub-processes; and the last three observers observe current authorizations defined on the three sub-processes with the purpose to check: currently, privilege to
execute some task has been granted to which role. And such a privilege is denoted by an authorization. For each workflow instance, there exist one authorization history of actual authorizations granted to roles and one execution history of actual executions of tasks by subjects. Authorization/Execution history is denoted by a queue, whose elements are authorization/execution records. During the execution of workflow, all authorizations that have been granted to roles (i.e. authorization records) are added to the the queue of authorization history, and all executions of tasks by subjects (i.e. execution records) are added to the queue of execution history.

5.3.3 Transparency Discussion of Financial Budget Disclosure

So the information flow of current financial budget disclosure model is one way report. To citizen, they do not have right to access the information they want by themselves. Intuitively speaking, the operation transparency and information transparency are not complete. Since the financial budget is a kind of special law document, there are in the model, if the citizen can engage in the budget making process, the regulatory transparency can be discussed.

5.4 Simulation Discussion

Before giving the concrete discussion on simulation between cases and general model, let’s first understand two questions: why we need to do this simulation work, i.e., the
motivation or what benefits we can obtain from this work, and how we did this work.

By the first question, (a) we proposed a generic model for e-Government, and then we wrote CafeOBJ specifications for this model. (b) Then we proved that the specifications have several desired (transparency) properties by doing induction. (c) Following the idea of the generic model, then wrote specifications for several real-life e-Government systems of social welfare, business licensing, and financial budget disclosure.

We want to prove that the specifications of the three systems have some desired properties. Should we need to prove them again by doing induction on their specifications? Is there any other means that might be easier? Therefore, we come to the answer “yes”, simulation is possibly an easier way to prove these properties.

The idea of the simulation technique based on the OTS/CafeOBJ method is as follows [49]: Suppose that (a) we have proved some properties, denoted $P_a$, hold for a system modeled as an OTS, denoted $S_a$, ($S_a$ is relatively less complicated, and $P_a$ can be easily proved for $S_a$), and (b) we have another system modeled as another OTS, denoted $S$. $S$ is more complicated than $S_a$. If we can find/prove that there is a simulation relation $r$ from $S$ to $S_a$, and that $P_a$ implies $P$ when assuming $r$ (hold), then we have proved $P$ without doing induction on $S$.

So how did we come up the idea to do simulation? The main reason is that: the three real-life systems are very similar to the proposed generic model (actually this is the essential reason that we propose this generic model). And besides, the properties of the three systems (that we want to prove) are also very similar to those properties of the generic model (that we have already proved). Then there is a high possibility that a certain simulation relation exists. That is why doing simulation proof is, at least seemingly, feasible.

The simulation proof indeed easier than induction proof (or more honestly, it will be easier. We can show this more convincingly after we finish the simulation proof for many properties. Currently we only proved one property). The proofs for the implication relation between properties are often simple, and the difficult work is to find/prove the simulation relation. Once we find such a simulation relation (proving this may be not quite easy, as shown by our work), we can prove many properties easily.

For carrying on the simulation proof, The first work we need to do is to find such a simulation relation $r$. This is done as follows: define a candidate relation $r$, and then prove $r$ is indeed a simulation relation from $S$ to $S_a$ by showing that $r$ satisfies the two conditions described in the following Definition which was presented in [49] Ogata and Futatsugi senseis paper.

Definition Given OTSs $S$ and $S_A$, $r$: $R_S \rightarrow R_{S_A}$ is called a simulation from $S$ to $S_A$ if it satisfies the following conditions:

(i) For each $v \in I$, there exists $v_A \in I_A$ such that $r(v, v_A)$.

(ii) For each $v,v' \in R_S$ and $v_A \in R_{S_A}$ such that $r(v, v_A)$ and $v \rightsquigarrow_s v'$, there exists $v'_A \in R_{S_A}$ such that $r(v', v'_A)$ and $v_A \rightsquigarrow_{s_A} v'_A$.

The simulation relation we proposed is as written by CafeOBJ, which is named as SW2GN-sim1. SW2GN-sim1 is defined in the file “simulation.mod”¹, and is proved to be a simulation relation in the file “sw2gn-sim1-proof.mod”. In this proof, five lemmas: “GN-lemma3”–“GN-lemmas7”, are used. The following equations are the excerpt of the

¹can be found in appendix
CafeOBJ code for describing the simulation relations between General Model and Social Welfare Model.

-- refinement relations
op SW2GN-sim1 : Sys1 Sys0 DocName -> Bool
-- definitions of the refinement relations
eq SW2GN-sim1(S1,S0,D)
= (docType(doc(S1,D)) = docType(doc(S0,D))) and
  (if (docLabel(doc(S1,D)) = appliedSW)
   then (docLabel(doc(S0,D)) = applied)
   else (if (docLabel(doc(S1,D)) = verified)
       then (docLabel(doc(S0,D)) = evaluated)
       else (if (docLabel(doc(S1,D)) = decided)
           then (docLabel(doc(S0,D)) = confirmed)
           else true)
     fi) fi) fi) fi) fi) fi) fi) fi) fi) fi) fi
.

After above work, then we need to do is to show that: the properties proved for the generic model can imply the properties of the three systems that we want to prove. (The correctness of this reasoning is shown as the following Theorem and proof can be found in paper [49]).

**Theorem** Given arbitrary OTSs $S$ and $S_A$ such that there exists a simulation from $S$ to $S_A$, an arbitrary such simulation $r$ and arbitrary state predicates $p, p_A$ such that $p_A(v_A) \Rightarrow p(v)$ for arbitrary states $v, v_A$ with $r(v, v_A)$, if $\forall v_A : R_{S_A}.p_A(v_A)$, then $\forall v : R_S.p(v)$.

The excerpt of CafeOBJ descriptions of invariants for general model and social welfare model are shown as follows:

-- Defining invariants for General model

\[
\eq{GN-inv1(S0,A,D)} = \begin{cases} 
[A,evaluate] /in \text{docTrace}(\text{doc}(S0,D)) \text{ or } [A,confirm] /in \text{docTrace}(\text{doc}(S0,D)) \\
\implies \neg([A,apply] /in \text{docTrace}(\text{doc}(S0,D))) 
\end{cases} .
\]

\[
\eq{GN-inv2(S0,A,D)} = \\
(A,execute) /in \text{docTrace}(\text{doc}(S0,D)) \implies \neg(A,apply) /in \text{docTrace}(\text{doc}(S0,D)) .
\]

\[
\eq{GN-inv3(S0,A1,A2,D)} = \\
(A1,confirm) /in \text{docTrace}(\text{doc}(S0,D)) \text{ and } [A2,execute] /in \text{docTrace}(\text{doc}(S0,D)) \\
\implies \neg(A1 = A2) .
\]

\[
\eq{GN-lemma1(S0,A,D,D)} = \\
(A,apply) /in \text{docTrace}(\text{doc}(S0,D)) \implies \text{docCreator}(\text{doc}(S0,D)) = A .
\]

-- for proof inv3

\[
\eq{GN-lemma2(S0,A1,A2,D)} = \\
(A1,execute) /in \text{docTrace}(\text{doc}(S0,D)) \implies [A2,confirm] /in \text{docTrace}(\text{doc}(S0,D)) .
\]

-- Defining invariants for Social welfare

\[
\eq{SW-inv1(S1,A,D)} = \\
(A,verify) /in \text{docTrace}(\text{doc}(S1,D)) \text{ or } [A,decide] /in \text{docTrace}(\text{doc}(S1,D)) \\
\implies \neg([A,apply] /in \text{docTrace}(\text{doc}(S1,D))) .
\]

\[
\eq{SW-inv2(S1,A,D)} = \\
(A,issue) /in \text{docTrace}(\text{doc}(S1,D)) \implies \neg([A,apply] /in \text{docTrace}(\text{doc}(S1,D))) .
\]

\[
\eq{SW-inv3(S1,A1,A2,D)} = \\
(A1,decide) /in \text{docTrace}(\text{doc}(S1,D)) \text{ and } [A2,issue] /in \text{docTrace}(\text{doc}(S1,D)) \\
\implies \neg(A1 = A2) .
\]
In this proof, six lemmas: “SW-lemma1” to “SW-lemma6”, are used \(^2\). Of course, we also need to prove all these lemmas so as to finish the whole simulation proof.

And by CafeOBJ proof engine, we proved the invariant holds, therefore it exists a simulation relation between social welfare case and general model. In our case study, we use an abstract protocol (GSD model which be mentioned in Ch.3), and three cases as examples. To proof refinement, we prove that there exists a simulation r1 from Social Welfare case model to GSD model, therefore the invariant which be proved simulation basic from transparency properties, can hold on the GSD’s simulation cases.

So if the case model has the simulation relation with GSD model, then the properties we have proved on GSD model can be easily proved on the simulation models. Therefore the case model can be proved to fulfill the transparency level 0 if there exists simulation relations between cases and GSD model, since we have proved the GSD model fulfill the transparency level 0.

5.5 Summary

![Figure 5.4: A Simple Compare between GSD Model and Cases](image)

In this chapter, we have described three case studies to check the usefulness and effectiveness of the methodology we proposed in chapter 4. The figure illustrates some simplified model and property discussion issues. The cases can be considered as a kind of middle level abstraction, and detailed information may not exactly same but has been refined from real government cases. Beyond the property which be held on the GSD model, we discussed more transparency properties description in this chapter, and analyzed in what kind of situation the properties can be held in the systems. Based on the discussion in this chapter, we provide some potential improvement suggestions to improve system transparency.

\(^2\)This step is written in the file “sw2gn-inv1-proof.mod”.
Chapter 6

A Public Government License Language

After transparency discussion in previous chapters, in this chapter, we attempt to propose a potential method for transparency improvement – Public Government License Language (PGLL). Intuitively speaking, the information flow changes from one way report model into two way communication after imported PGLL idea, therefore it has high probability on improving the information transparency which we described in chapter 4.

So what is a license? By a license: A right or permission granted in accordance with law by a competent authority to engage in some businesses or occupations, to do some acts, or to engage in some transactions which but for such license would be unlawful – Merriam Webster Online. In the rest of this chapter, we propose the concept of license, the syntax and semantics of PGLL, and the experiments we did on public cases discussion by PGLL.

6.1 Concept of License

6.1.1 A Concept of License

The concepts of licenses and licensing express relations between actors (licensors – the authority and licensees), entities (works, patients, documents) and functions (on entities), and as performed by actors. By issuing a license to a licensee, a licensor wishes to express and enforce certain permissions and obligations: which functions on which entities the licensee is allowed (is licensed, is permitted) to perform.

Here we consider entity as the domain understanding in chapter 2: document, say of public government. And for public administrators and citizens to enjoy good governance: transparency in law making (national parliaments and local prefecture and city councils), in law enforcement (i.e., the daily administration of laws), and law interpretation (the judiciary) — by agents who are basically obliged to produce certain documents while being permitted to consult (i.e., read) other documents.

After we have now understand the Public Administration Domain, we shall design a suitably flexible public government document license language and, through its precise informal and formal description provide one set answers to the above issue and show some case studies in this area, in the rest part of the report.

Before starting design the PGLL, we shall take a closer look at the “permission and
obligation” in Public Administration, that are addressed in, i.e., which are expressed in licenses. To give a first idea of a license, as a syntactic entity, let us postulate a rather simple-minded license:

\[
\text{license } l : \text{licensor } \alpha \text{ grants licensee } \beta \text{ work}\omega
\]

\[
\text{with permitted actions } \{p^{a_1}, p^{a_2}, ... p^{a_n}\}
\]

\[
\text{and obligated actions } \{o^{a_1}, o^{a_2}, ... o^{a_n}\}
\]

The above license express: if licensor issues license \(l\) on work \(\omega\) to licensee \(\beta\). So the \(\beta\) is allowed (permitted) to perform action \(\{p^{a_1}, p^{a_2}, ... p^{a_n}\}\) on \(\omega\) while obligate (force to) perform action \(\{o^{a_1}, o^{a_2}, ... o^{a_n}\}\).

6.1.2 Concept of License Language

We distinguish between the semantics and the syntax of that language. Leading textbooks on (formal) semantics of programming languages are

- **By syntax** we mean (i) the ways in which words are arranged to show meaning (cf. semantics) within and between sentences, and (ii) rules for forming syntactically correct sentences.

- **By semantics** we mean the study and knowledge [incl. specification] of meaning in language [6]. We make the distinction between the pragmatics, the semantics and the syntax of languages. Leading textbooks on (formal) semantics of programming languages are [7,8,9,10,11,12]. By informal semantics we mean a semantics which is expressed in concise natural language, for example, as here, English.

So before enter the topic of PGLL and how to apply it in public administration domain, we should design the concreted syntax and semantics about it firstly. In public administration domain: the entities of PGLL are the licensors, licensees, license \(l\) and the works \(\beta\). We focus on \(l\) and \(\omega\). We consider licensors and licensee as behavior.

From the rough syntax of License Language, we can find out that there are two parties to a license: the licensor and the licensee. And there is a common agreement concerning a shared “item” between them, namely: the license \(l\) and the document \(\omega\).

- The licensor gives the licensee permission, or mandates the licensee to be obligated to perform certain actions on designated “items”.

- The licensee performs, or does not perform permitted and/or obligated actions. And the licensee may perform actions not permitted or not obligated.

The license shall serve to ensure that only permitted actions are carried out, and to ensure that all obligated actions are carried out. Breach of the above, that is, breach of the contracted license may or shall result in revocation of the license.

**Licensor in PGLL**

Licensor, as we analyzed above, who can authorize a license to other actors, should make sure the license they issue with different name. If a licensor has authorised a licensee to
create a document (and that document, when created got the unique document identifier udi:UDI) then that licensee can release the created, and possibly edited document (by that identification) to the licensor, say, for comments. The licensor thus obtains the master copy.

Licensee in PGLL
A licensee can sublicense certain operations to be performed by other actors. The granting, extending, restricting or withdrawing permissions, cannot name a license (the user has to do that), do not need to refer to the licensor (the licensee issuing the sublicense), and leaves it open to the licensor to freely choose a licensee. One could, instead, for example, constrain the licensor to choose from a certain class of actors.

Document in PGLL
Documents: A crucial means of expressing public administration is through documents. We have provided a brief domain analysis of a concept of documents in chapter 2. (This document domain description also to artistic works insofar as they also are documents.)

Operations on Document
Documents to us, at this point in the report, are for the analyzed values of some sort. When we, in the next section 5.3, deal with operations in PGLL, then we shall endow/enclude document with the operations that can be performed. Documents Documents are created, edited and read; and documents can be, distributed, the subject of calculations (interpretations) and be shared and shredded. With documents one can associate, as attributes of documents, the actors who created, edited, read, distributed and (to whom distributed), shared performed calculations and who on, shredded documents.

Document Attributes
With these operations on documents, and hence as attributes of documents one can, again conceptually, associate the location and time of these operations.

Document Tracing
An issue of public government is whether citizens and agents of public government act in accordance with the laws with actions and laws reflected in documents such that the action documents enables a trace from the actions to the laws “governing” these actions. We shall therefore assume that every document can be traced back to its law origin as well as to all the documents any one document creation or editing was based on.

Licenses and Actions
Conceptually a licensor o (for owner) may issue a license named ℓ to licensee u (for user) to perform some actions. The license may syntactically appear as follows: ℓ : licensor o licenses licensee u

---

1Documents are, for the case of public government to be the “equivalent” of artistic works.
to perform actions a1,a2,..an on work item w Actions
And, conceptually, the licensee (u) may perform actions ℓ :a(w); ℓ :a(w); ℓ :a (w);
ℓ :a(w)
There are two kinds of actions on documents: create, edit, read, share, shred, copy
are on kind performed by licensee, and licensor issue license ℓ to others, is performed on
license.

License Library in PGLL

With actors (whether agents of public government or citizens) one can associate the au-
thority (i.e., the rights) these actors have with respect to performing actions on documents.
We now intend to express these authorizations as licenses. License library can save some
common license which may often use in public administration domain and improve the
executing efficiency.

6.2 Public Government License Language

6.2.1 Syntax Licenses

For testing the effectiveness of PGLL, we postulate a draft PGLL by a BNF grammar
with start symbol L; given rules for what we can replace this symbol with and the PGLL
language defined by the BNF grammar is just the set of all strings you can produce by
following these rules. (The test can be found in the next section).

Here we use Ln, An, Doc, Dn, Cfn instead for License name, Actor name, Document,
Document name and Calculate function respectively.

1. \( L :: = Grant | Extend | Restrict | Withdraw \)

2. Grant :: = Ln An grants An Ops
   Extend :: = An extends An Ln Ops

3. Restrict :: = An restricts An Ln Ops
   Withdraw :: = An withdraws An Ln

4. Ops :: = Op | Op Ops

5. Op :: = Create | Edit | Read | Copy | Share | Revoke | Calculate | Shred | License

6. Create :: = create D based on D

7. Read :: = read D

8. Edit :: = edit D into D

9. Copy :: = copy D into D

10. Share :: = share Dn Ans

11. Revoke :: = revoke Dn Ans

12. Calculate :: = calculate Cfn D yielding D

82
13. Shred :: = shred Dn
14. License :: = MandatedD | MandatedL
15. MandatedD :: = do Ops
16. MandatedL :: = lic Ops into Ans
17. Ans :: = An | An Ans
18. Cfns :: = Cfn | Cfn Cfns

6.2.2 Annotations : License

So a Public Government License consists of either Grant or Extend or Restrict or Withdraw part.

Grant part consists of License name(Ln), one Actor name(An) as a licensor, Operation set(Ops) and another Actor name(An) as a licensee.

Extent part consists of first An as licensor who extend license which gave by himself before and second An as licensee, following by License name part and operation set part which be extended in license.

Restrict part consists of first An as licensor who restrict license and second An as licensee, following by License name and operation set part which will drop out from license.

Withdraw part consists of first An as licensor and second An as licensee, following by License name which going to be withdraw.

Operation set part consists of combination with Create, Edit, Read, Copy, License, Share, Revoke, Calculate and Shred.

Create part consists of first Document name part as new document and second document name part as which the new document generated based on.

Read part consists of only the document name which be read.

Edit part consists of document name and same name after edit.

Copy part consists of document name part. And copying a document increases the document population by exactly one document. All previously existing documents remain unchanged except that the document which served as a master for the copy has been so marked. The copied document is like the master document except that the copied document is marked to be a copy (etc.). The master document, if not the result of a create of copy, is moved from licensor to licensee – if not already so moved as the result of the specification of other authorized actions.

Share part consists of Document name which be shared and Actors name who shared the document.

Revoke part consists of Document name part and actors name part.

Calculate part consists of Calculation function set and Document name which the calculate will be applied on.

Shred part consists of document name which need be shred.

Op without License means can either do it by agent itself or License other agent to do it. So operations with License here means either License can only authorize to other to do (MandatedL) or can only do by actor itself (MandatedD)
(1) There are names of licenses (Ln), actors (An), documents (Dn), document classes (DCn) and calculation functions (Cfn).
(2) There are four kinds of licenses: outright, initial licensing, (license extensions, license restrictions and license withdrawals)
(3) Actors grant licenses. An actor is constrained to always grant distinctly named licenses. No two actors grant identically named licenses. A set of operations on (named) documents are granted.
(4-6) Actors grant who have issued named licenses may extend, restrict or withdraw the license rights (wrt. operations, or fully).
(7) There are nine kinds of operation authorizations.
Some of the explications below also explain parties of some of the corresponding actions (see (16-24))
(8) Creation results in a document identified by a given Dn and referring to a named document class, DCn. Creation of an initially void document may be based on one or more identified documents (over which the licensee (at least) has reading rights). What this ”be based on” means is simply that the initially void document contains references to those document. (They can therefore be traced (etc.) - as per [18].) The ”based on” documents are moved from licensor to licensee.
(9) Editing a document may be based on ”inspiration” from, that is, with reference to a number of other documents (over which the licensee (at least) has reading rights). What this ”be based on” means is simply that the edited document contains those references.(They can therefore be traced.) The ”based on” documents are moved from licensor to licensee - if not already so moved as the result of the specification of other authorized actions.
(10) Reading a document only changes its ”having been read” status (etc.) - as per [18]. The read document, if not the result of a copy, is moved from licensor to licensee - if not already so moved as the result of the specification of other authorized actions.
(11) Copying a document increases the document population by exactly one document. All previously existing documents remain unchanged except that the document which served as a master for the copy has been so marked. The copied document is like the master document except that the copied document is marked to be a copy(etc.) -as per The master document, if not the result of a create and edit or copy, is moved from licensor to licensee - if not already so moved as the result of the specification of other authorized actions.
(12) Op without License means can either do it by agent itself or License other agent to do it. So operations with License here means either License can only authorize to other to do (MandatedL)or can only do by agent itself (MandatedD)
(13) A document can be shared between two or more actors. One of these is the licensee, the others are implicitly given read authorizations. (One could think of extending, instead the licensing actions with a shared attribute.) The share document, if not the result of a create and edit or copy, is moved from licensor to licensee - if not already so moved as the result of the specification of other authorized actions. Sharing a document does not move or copy it.
(14) Sharing documents can be revoked. That is, the reading rights are removed.
(15) Two or more documents can be subjected to any one of a set of named calculation functions. These documents, if not the result of a creates and edits or copies, are moved from licensor to licensee - if not already so moved as the result of the specification of other
authorized actions.

(16) A document can be shredded. It seems pointless to shred a document if that was the only right granted wrt. document.

### 6.2.3 Syntax: Actions

17. Action :: = Ln Cre | Edt | Rea | Cop | Lic | Sha | Rev | Cal | Shr
18. Cre :: = Dn DCn Dn-set
19. Edt :: = Dn Dn-set
20. Rea :: = Dn
21. Cop :: = Dn Dn
22. Lic :: = MandatedD | MandatedL
23. Sha :: = Dn An-set
24. Rev :: = Dn An-set
25. Cal :: = Cfn Dn-set Dn
26. Shr :: = Dn

### 6.2.4 Annotations: Actions

(17) Actions are tagged by the license with respect to which their authorisation and document names has to be checked. There are nine actions. No action can be performed by a licensee unless it is so authorised by the named license, both as concerns the operation (create, edit, read, copy, license, share, revoke, calculate and shred) and the documents actually named in the action. They must have been mentioned in the license, or, created or copies of downloaded (and possibly edited) documents or copies of these - in which cases operations are inherited.

(18) A licensee may (conditionally) create documents as so licensed and obtains all operation authorizations to this document.

(19) A licensee may (conditionally) edit “downloaded” (edited and/or copied) or created documents.

(20) A licensee may (conditionally) read “downloaded” (edited and/or copied) or created and edited documents.

(21) A licensee may (conditionally) copy “downloaded” (edited and/or copied) or created documents. The licensee decides which name to give the new document, i.e., the copy. All rights of the master are inherited to the copy.

(22) A licensee can either do the job themselves or give the job to others or receive a job that it can only do by itself.

(23) A “downloaded” document may be shared with one or more other actors. Sharing, in a digital world, or example, means that any edits done after the opening of the sharing session, can be read by all so-granted other actors.

(24) Sharing may be revoked, partially or fully, that is, wrt. original “shareers”.

(25) A license may apply any of a licensed set of calculation functions to “downloaded” documents, or can apply any of a licensed set of calculation functions to created documents. The result of a calculation is a document. The license decides which name to give this document. The licensee obtains all operation authorizations to this document.

(26) A license may (conditionally) shred a “downloaded” (etc.) document.
6.3 Experiments with PGLL

We now subject the draft PGLL syntax of Ch.6 to a test: namely whether using only this syntax we can express what we consider important license. Those licenses are now exemplified there below to four areas. (i) G2G: Financial Budget; (ii) G2B: Company Registration; (iii) G2C: Income Tax.

6.3.1 Experiments with Case Studies

Financial Budget

![Figure 6.1: Financial Budget](image)

**Description of Financial Budget** Government Financial want to decide the financial budget of next year. It gives permission to the central administration and asks them to submit corresponding budget appeal document. Then the central administration grants the local administrations. The local administrations submit their financial plans and revise them by advice from central administration. After decided, the central administration submits the final plan to government financial. And the budget plan will be revised by the advice from government financial. At last, government financial gives the financial budget.

**Natural Language License Description**
1) Government Financial authorizes the central administration section to handle the issues about appealing financial budget from local administrations. 2) Central administration section asks the local administrations to submit appealing forms for their financial budgets. 3) Local administrations submit the draft about their budgets and let central government to checkup. 4) Central administration gives its’ advice about the budget draft and let local administrations to revise and submit. 5) Local administrations submit their finance appeal documents and authorize central administration to gather the information and submit to government financial. 6) Central administration submits the collection report and allows government financial to make financial budget from the report. 7) Government Financial authorize the central administration to publish the final financial budget to local administrations.

**Description by PGLL**

\[
\mathcal{L} ::= \text{handle-License} \quad \text{GF grants} \quad \text{CA}
\]
read AF FB
copy FB into FB
lic create AF based on FB
  edit AF
  copy AF into AF
  shred AF
  into LA
(2) appeal-License
\[ \mathcal{L} ::= \text{appeal-License CA grants LA} \]
read AF
  edit AF into AF
  copy AF into AF
  do create AF based on FB
  shred AF
(3) submit-License
\[ \mathcal{L} ::= \text{submit-License LA grants CA} \]
read FA RD
  copy FA RD into FA
  do create FA RD based on AF
  edit FA RD
  shred FA RD
(4) revise-License
\[ \mathcal{L} ::= \text{revise-License CA grants LA} \]
read RA FA RD
  copy RA FA RD into FA RD
  do create RA based on FA RD
  edit RA
  shred RA
(5) submit-License
\[ \mathcal{L} ::= \text{submit-License LA grants CA} \]
read FA CR
  copy FA CR into FA CR
  do create CR based on RA
  edit CR
  shred CR
(6) use-License
\[ \mathcal{L} ::= \text{use-License CA grants GF} \]
read DR FA CR
  copy DR FA CR into DR FA CR
  do create DR based on FA CR
  edit DR
  shred DR
(7) public-License
\[ \mathcal{L} ::= \text{public-License GF grants CA} \]
read DR
  copy RD into DR
Description of Company Registration

If an entrepreneur who wants to found a company, he will have to go to business administration, and find a bank to get capital certification document and submit corresponding document. And business administration section will judge these documents by the corresponding law which published by financial ministry. The company will get business license if fit the conditions of business administration sections. So that the process done.

Natural Language License Description 1) Financial ministry authorizes the business administration dealing with the corresponding issues about company registration by corresponding company law.
2) Business administration asks the company who want to do registration by filling corresponding documents. Business administration let the company to choose the bank which can evaluate its’ capital.
3) Company chooses the bank to evaluate its’ capital.
4) Bank gives the capital certification to the company.
5) Company withdraws the capital certification license from bank.
6) Company submit documents to business administration.
7) Business administration restricts the company’s rights to read and copy the application documents they submitted after the company got business license.
8) Company withdraw the right of business administration to only archive the company information.
9) Business administration submit corresponding document to Financial Ministry.

Description by PGLL

(1)executive-License
$L ::= \text{executive-License FM grants BA}$
read R&R
$\text{do create R&R based on CL}$
edit R&R
shred R&R
(2) apply-License
\[ L ::= \text{apply-License BA grants Company} \]
  \begin{align*}
    & \text{read CC AD CI} \\
    & \text{copy CC AD CI into CC AD CI} \\
    & \text{do create AD CI based on R&R} \\
    & \text{edit AD CI} \\
    & \text{shred AD CI} \\
    & \text{lic create CC based on CL} \\
    & \text{edit CC} \\
    & \text{shred CC} \\
    & \text{into Bank} \\
  \end{align*}

(3) certification-License
\[ L ::= \text{certification-License Company grants Bank} \]
  \begin{align*}
    & \text{read CC} \\
    & \text{copy CC into CC} \\
    & \text{do create CC based on CL R&R} \\
    & \text{edit CC} \\
    & \text{shred CC} \\
  \end{align*}

(4) submit-License
\[ L ::= \text{submit-License Bank grants Company} \]
  \begin{align*}
    & \text{read CC} \\
    & \text{copy CC into CC} \\
  \end{align*}

(5) withdraw-License
\[ L ::= \text{withdraw-License Company withdraws Bank certification-License} \]

(6) check-License
\[ L ::= \text{check-License Company grants BA} \]
  \begin{align*}
    & \text{read CC CI AD} \\
    & \text{copy CC CI AD into CC CI AD} \\
    & \text{do create BL based on CC CI AD} \\
    & \text{edit BL} \\
    & \text{shred BL} \\
  \end{align*}

(7) use-License
\[ L ::= \text{use-License BA grants Company} \]
  \begin{align*}
    & \text{read BL} \\
    & \text{copy BL into BL} \\
  \end{align*}

(8) archive-License
\[ L ::= \text{archive-License Company extends check-License BA} \]
  \begin{align*}
    & \text{read Archive} \\
    & \text{copy Archive into Archive} \\
    & \text{do create Archive based on CC CI AD} \\
    & \text{edit Archive} \\
    & \text{shred Archive} \\
  \end{align*}

(9) checkup-License
\[ L ::= \text{checkup-License BA grands FM} \]
  \begin{align*}
    & \text{read Archive CR} \\
    & \text{copy Archive CR into Archive CR} \\
    & \text{do create CR based on CC CI AD Archive} \\
  \end{align*}
Government Financial Report

Figure 6.3: Accounting Report

**Description of Accounting Report** Company records their daily business in accounting reports and reports it per periods. Audit checks if the company report correctly. And revenue takes in tax by the accounting reports and audit reports.

**Natural Language License Description**
1) Company shows its’ accounting report to public after an accounting period.
2) Audit be given right to check accounting report of the company Company and report result to government.
3) Government give withdraw the right audit had after finish audit procedure.
4) Audit gives corresponding result to company.
5) Company submit corresponding document to Revenue for paying tax.
6) Revenue gives the result back to company.

**Description by PGLL**

(1) audit-License

\[ L ::= \text{audit-License Company grants Audit} \]
- read AcR AR AB V
- copy AcR AR AB V into AcR AR AB V
- do create AR based on AR RkR
- edit AR
- shred AR

(2) submit-License

\[ L ::= \text{submit-License Audit grants Government} \]
- read AR DR
- copy AR DR into AR DR
- do create DR based on AR
- edit DR
- shred DR

(3) publish-License

\[ L ::= \text{publish-License Government grants Audit} \]
- read DR
- copy DR into DR

(4) knowing-License

\[ L ::= \text{knowing-License Audit grants Company} \]
- read DR
- copy DR into DR

90
6.3.2 CafeOBJ Specification with PGLL

Syntax: License

Actors in PGLL are divided into Licensor and Licensee, we use the following CafeOBJ description to declare it.

\[
\begin{align*}
\text{mod! LICENSEE} \\
\quad \text{pr}(\text{NM}) \\
\quad \quad \text{[Licensee]} \\
\quad \quad \text{op leename : Licensee} \rightarrow \text{Nm} \\
\text{mod! LICENSOR} \\
\quad \text{pr}(\text{NM}) \\
\quad \quad \text{[Licensor]} \\
\quad \quad \text{op lorname : Licensor} \rightarrow \text{Nm} \\
\text{mod! ACTOR} \\
\quad \text{pr (LICENSEE + LICENSOR)}
\end{align*}
\]

Distinguish with the actor we defined in public administration domain, here the LICENSEE and LICENSOR sorts constitute the ACTOR sort in this domain. As we already mentioned above, the licensor is who give out the license and the licensee is who received the license. We try to map our CafeOBJ Specification of PGLL to Chap.6, so that our CafeOBJ specification will be more soundable. Corresponding to the syntax part of PGLL, we give our CafeOBJ description as following:

\[
\begin{align*}
\text{op lnm : Olicense} \rightarrow \text{Ln} & \quad \text{-- get license name} \\
\text{op lor : Licensor} \rightarrow \text{Nm} & \quad \text{-- get licensor name} \\
\text{op lee : Licensee} \rightarrow \text{Nm} & \quad \text{-- get licensee name} \\
\text{op acs : Olicense} \rightarrow \text{Actionset} & \quad \text{-- get action permission in license} \\
\text{op dn : Doc} \rightarrow \text{String} & \quad \text{-- document name}
\end{align*}
\]

Here we use \text{lmm}, \text{lor} and \text{lee} to represent license's name, licensor's name and licensee's name respectively.

Since we distinguish License into four kinds of licenses in previous section.

\[
\begin{align*}
\text{pr ( OLICENSE + ELICENSE + RLICENSE + WLICENSE )} \\
\text{pr ( EQL )} \\
\quad \text{[ Olic Elic Rlic Wlic < License]} \\
\text{ops olic elic rlic wlic :} \rightarrow \text{License} \\
\text{op obsLicType : License} \rightarrow \text{License} \\
\text{var O : Olic} \\
\text{var E : Elic} \\
\text{var R : Rlic}
\end{align*}
\]
var W : Wlic
eq obsLicType (O) = olic .
eq obsLicType (E) = elic .
eq obsLicType (R) = rlic .
eq obsLicType (W) = wlic .

**OLICENSE, ELICENSE, RLICENSE** and **WLICENSE** refer to original license which key word is *grants* in PGLL, extend license which key word is *extends*, restrict license which key word is *restricts* and withdraw license which key word is *withdraw*. And the operator obsLicType is the operator in CafeOBJ definition to get license type.

*grants* license is original license, which present the basic permission and obligation relations between licensor and licensee;
*extends* license base on grants license, which extend the basic right of grant license;
*restricts* license also base on grants license, and restrict some right from basic license;
*withdraw* license, withdraw grants license, which means revoke all permission and obligation in one license.

**Semantic : License**

To define the above four licenses in details. We declare **original license** as following.

**mod! OLICENSE**

```
pr ( ACTOR + ACTIONSET + LN + DOC + EQL )
[Olicense]
op lnm : Olicense -> Ln -- get license name
op lor : Licensor -> Nm -- get licensor name
op lee : Licensee -> Nm -- get licensee name
op acs : Olicense -> Actionset -- get action permission in license
op dn : Doc -> String
```

And we define extends license by adding actions in action set of original license.

**mod! ELICENSE**

```
pr ( OLICENSE + ACTIONSET + EQL )
[Elicense]
op eorg : Elicense -> Olicense
op elnm : Elicense -> Ln
op elor : Elicense -> Nm
op elee : Elicense -> Nm
op aacs : Elicense -> Actionset -- actions which are added in Extend License
op eacs : Elicense -> Actionset
var O : Olicense
var E : Elicense
vars AS AS' : Actionset
ceq elnm(E) = lnm(O) if eorg(E) = O .
ceq elor(E) = lor(O) if eorg(E) = O .
ceq elee(E) = lee(O) if eorg(E) = O .
ceq eacs(E) = add(aacs(E), acs(O)) if eorg(E) = O .
```

And we define restrict license by subtracting actions in action set of original license.

92
mod! RLICENSE
pr ( OLICENSE + ACTIONSET)
[Rlicense]
op rorg : Rlicense -> Olicense
op rlnm : Rlicense -> Ln
op rlor : Rlicense ->Nm
op rlee : Rlicense ->Nm
op sacs : Rlicense -> Actionset -- actions which are added in Extend License
op racs : Rlicense -> Actionset
var O : Olicense
var R : Rlicense
vars AS AS' : Actionset
ceq rlnm(R) = lnm(O) if rorg ( R ) = 0 .
ceq rlor(R) = lor(O) if rorg ( R ) = 0 .
ceq rlee(R) = lee(O) if rorg ( R ) = 0 .
ceq racs(R) = sub(ac(O), sacs( R )) if rorg ( R ) = 0 .

And we define withdraw license by making the action set empty.

mod! WLICENSE
pr ( OLICENSE )
[Wlicense]
op worg : Wlicense -> Olicense
op wlnm : Wlicense -> Ln
op wlor : Wlicense ->Nm
op wlee : Wlicense ->Nm
op wacs : Wlicense -> emptya
var O : Olicense
var W : Wlicense
vars AS AS' : Actionset
ceq wlnm(W) = lnm(O) if eorg ( W ) = 0 .
ceq wlor(W) = lor(O) if eorg ( W ) = 0 .
ceq wlee(W) = lee(O) if eorg ( W ) = 0 .
ceq wacs(W) = emptya if worg ( W ) = 0 .

Annotations: Actions
We associate time and operations on document to define Action in PGLL.

mod! TIME
pr (NAT)
[Time]

mod! ACTION
pr(EQL + STRING + TIME + EDITING + LEESET + FUNCTION)
[Action]
op create : String Nat -> Action
op edit : Editing Nat -> Action
op read : Nat -> Action
op copy : Nat Nat -> Action
op mdlic : Nat -> Action
op dolic : Nat -> Action
op share : Leeset Doc -> Action
op calu : Function Doc -> Action
op shred : Doc -> Action
The above part is the basic part of actions. The argument with corresponding operators refers to: String in create refers to create document named; by the string edit refers to do what kind of editing; read changes nothing; copy with Nat refers to number of copies; mdlic refers to mandate license which means the operations can do only by license others to do; dolic refers to do license which means the operations can do only by the licensee itself; share with licensee set and times; calu refers to perform a function on corresponding document; shred document(Doc).

Licensed Behaviors

The following figure illustrates which stages corresponds to what kinds of operations on each document. By analyzing PGLL, we want to analyze if all the operations are well-defined.

6.4 Summary

In the this chapter, we proposed a license language for public government (PGLL), and gave syntax, semantics and formal description by OTS/CafeOBJ as well. Then we applied PGLL into the public administration cases which also corresponding to the cases studies.
we mentioned in chapter 5, to test the effective of the PGLL we proposed. Intuitively speaking, the information flow changes from one way report model into two way communication after imported PGLL idea, therefore it has high probability on improving the information transparency which we described in chapter 4. And more concreted formal discussion on this issues is an ongoing work, we are planning to do in the near future.
Chapter 7
Conclusions

7.1 Summarization

This thesis concerns how to define, evaluate and embed desired property (especially, transparency property, here) into e-Government systems from formal method and software engineering’s point of view. Two main issues discussed in the thesis are first methodology – how to use domain engineering, formal method and software engineering methodology to look into the real world and combine engineering development with the theory; and second property description – how to formally describe and analysis transparency, a kind of social science property, from a formal method point of view.

The specific combination of the OTS/CafeOBJ method and Domain engineering is used as a tool to understand, model and analyze the real world systems. The main reason of selecting OTS/CafeOBJ methods is that: a number of already published case studies has shown the usability and effectiveness of the inductive verification technique for proving invariant properties and have chance to be applied into real software development. And the main reason of selecting domain engineering is before we get into the real system design and discussion we need understand the application domain of it.

Methodology – Refreshing the Traditional Social Method

By introducing the ICT in public administration and public service delivery, e-Government provides technology support for more effective and efficient public service delivery on one hand, and it reforms the traditional organization of public administration on the other hand. Since e-Government has closely connected with software development however many desired properties in traditional social science are difficult to be settle down at the beginning stage of a software development, even it is not easy to evaluate after the system has been established. Traditional social science methodology, which usually test system performance afterward by questionnaire, empirical or statistical data and feedback to revise the system, of course is one effective method for establishing well performance system, however it is not good for software engineers and other stake-holders to control the risk and cost of the project. To achieve this end, we broke the traditional social methodology and imported the domain engineering and formal method to coordinate the desired property in social science with the software system development.

For this issue, our methodology can be summarized as follows:

- We first proposed a general service delivery (GSD) model;
And then we gave some definitions on government system transparency, as well as classified those definitions into 4 different levels, system which fulfill the descriptions is considered as achieving corresponding transparent level;

After above, we analyze that GSD model can achieve the basic transparent level, so that the system which develop following GSD model can be considered as a basic transparent system;

In the case study, we added more detailed descriptions into GSD model; then we did the refinement discussion between the GSD model and cases, to show the cases have already fulfill the basic transparency;

Then we analyze more properties based on each concreted cases, and illustrate by giving more constraints to the systems based on the different cases, we can achieve more high level transparent systems.

As we present in previous chapter, our method is surely not panacea for solving government system transparency problems; however, we are providing a constructive suggestion from views of software engineering for attacking some social properties by engineering method and supporting to realize a more transparent e-Government systems.

Property Description – Formal Method between Social Theory and Software Theory

For the property description, since it is a long time controversial issue, different areas have different understanding. So, though it has been widely used, transparency has been rarely defined in a precise way. In natural science, transparency is used for describing materials’ property of condition for transmitting light to the outside or vice versa. In public administration field, transparency refers that information is freely available and directly accessible to those who will be affected by decisions and that enough information is provided in easily understandable forms and media.

Even there has some common knowledge on the concepts of transparency; different researchers still have different understanding depending on their ways to discuss it. In some papers, transparency has been understood as constituting a layman’s basic map of the organization as depicted in the information on the site and reveals the depth of access it allows, the depths of knowledge about processes it is willing to reveal, and the level of attention to citizen response it provides, in his evaluation on the citizen satisfaction of government web-service. In tracking financial resource and the identification of process features, transparency has been identical as openness of financial allocations, cost estimates and expenditures to stay in the public domain. In government fiscal study, transparency be quantitatively transferred into the share of the control-error variance arising from the observed component, under which the public perfectly infers the bank’s intention each period. The process should be transparent so that the public can see what is going on and how decisions are being made.

As in many other researches, we employed more metaphorical meanings of transparency than process it directly. It may not be able to stratify all different tasted readers, but we tried to cover the description as all-around as possible.
7.2 Related Work

This research is considered as a ground-breaking work, most of related work is attacking the transparent related issues instead of discussion transparency directly. Therefore we broke the related work into the following three viewpoints:

**Public Administration Viewpoint – Anti-Corruption**

In public administration area, transparency usually is considered as an effective strategy to against corruption, vise verse, a government system which contains less corruption behavior can be considered as a kind of transparent system. There are rare material go with transparency issue directly, but while mentioned anti-corruption, there are many researchers and organizations are working on it. Here we look into famous two in them. One is Transparency International and the other is World Bank [29].

Transparency International (TI) is an international non-governmental organization addressing corruption. It includes, but is not limited to, political corruption. It is widely known for producing its annual Corruptions Perception Index (see below), a comparative listing of corruption worldwide. The international headquarters is located in Berlin, Germany. TI publishes a report for evaluating country transparent by statistic data they gathered each year.

The Executive Board of the World Bank approved its anti-corruption strategy in September 1997. The strategy defined corruption as the “use of public office for private gain” and called for the Bank to address corruption along four dimensions: (i) Preventing fraud and corruption in Bank projects; (ii) Helping countries that request Bank assistance for corruption; (iii) Mainstreaming a concern for corruption in Bank’s work; and (iv) Lending active support to international efforts to address corruption. Following the adoption of this strategy, concrete steps to prevent fraud and corruption in Bank projects have included: the introduction of a confidential hotline, tightening of procurement guidelines, intensive audits of projects, and support for improving procurement systems in client countries. Mainstreaming a concern for corruption has taken place through the Bank’s economic and sector work which provides the analytic basis for country assistance strategy documents which, in turn, underlie the Bank’s lending programs.

However quantification of both transparency and corruption are not easy tasks. The menu of possible actions to contain corruption is very large, so Shah [8] and et al. develop a framework to help assign priorities, depending on views of what does and does not work in specific countries. Their framework, based on public officials’ incentives for opportunistic behavior, distinguishes between highly corrupt and largely corruption-free societies.

The Comprehensive Development Framework unveiled in 1999 underscores the importance of governance and public sector institutional reform issues in the Bank’s development assistance dialogue. The World Bank Institute has initiated courses on corruption for developing country officials and conducted surveys on service delivery. The Bank has also put a greater emphasis on institutional reform and capacity building through its lending program. Bank assistance for interactional efforts to curtail corruption has been mainly in the form of sponsorship of major conferences, dissemination notes and support for the adoption of the 1999 OECD Convention on Combating Bribery of Foreign Public Officials in International Business Transactions. The impact of corruption on public service delivery performance and poverty alleviation is widely recognized (see e.g. Tomaszewska and Shah, 2000 for empirical evidence). A wide consensus has also recently emerged that
corruption is a symptom of failed governance (see World Bank 2000) and hence curtailing corruption requires addressing the causes of mis-governance. Nevertheless, the menu of potential actions to curtail corruption is very large so a framework is needed that provides guidance on ordering potential actions. Prioritization of various actions depends on both the conceptual and empirical views of what works and what does not work in the context of particular countries. Such a framework is also needed for evaluating both Bank and country anti-corruption programs. This note proposes a framework for such evaluations. But there is rare effective framework to tutor e-Government system development.

**Information Technology Viewpoint – Information Accessibility**

Information accessibility is also not a new world in research area; it has been discussed for along time. Information accessibility is considered as a very important property in fair and transparent market.

And by development of information and communication technologies, discussion between information access and private protect has been last for a long time. To well define the information accessibility, some organization even proposed more detailed description of information accessibility. For instant, it’s described as the ability to access information and services by minimizing the barriers of distance and cost as well as the usability of the interface. Web Accessibility Initiative (WAI) develops the Web Content Accessibility Guidelines (WCAG) by separating the accessibility into 3 levels.

And by the appearing of search engines, some researchers propose information accessibility performance evaluation based on search engine performance. Because search engines do not index sites equally, may not index new pages for months, and no engine indexes more than about 16% of the Web. As the Web becomes a major communications medium, the data on it must be made more accessible [31].

However above researches are too general to have no clear target on discussing transparent issues against government web site transparency especially.

**Formal Technology Viewpoint – E-Government Construction Supporting**

The requirements of Electronic Governance extend strictly beyond those of the business world, in terms of flexibility, transparency and inter-operability, and the appropriate commercial solutions do not exist [34]. The demands of commerce will not produce the technology required; instead, government must drive its development – as an intelligent customer, and as an active partner in the design of e-Government solutions.

To drive the development of the technology, it is essential that governments, or their agents, be able to produce precise specifications of functionality. This kind of specification requires an approach to modeling known as formal engineering techniques: the use of mathematically-based languages, methods, and tools. Informal, imprecise use of graphical notations such as flow diagrams is not enough: the meaning of the specification must be clear and irrefutable.

Now the formal techniques for electronic government are mainly focus on discussion: inter-operability between collaboration parties can be enhanced through formal description of entities, events and actions involved in the interaction context; contract specifications in outsourced development projects by the IT function can benefit from clear formal specification of the desired system, particularly when specifying privacy and anonymity properties; verification of sources delivered by third-parties or suppliers can be carried
out using test cases generated from formal requirements specifications; detailed behavioral specification can be provided, and formal verification can be carried out to ensure privacy and anonymity properties are not violated in the supplied implementation.

However, the current researches are mainly limit to the technical topics rather than interdisciplinary discussion of changes and chance for reforms around government importing ICT.

7.3 Future Work

There is still a lot of work to be done towards making our already accomplished work complete. And the work is going to be carried out in the following three aspects:

Methodology

We illustrate in which way the domain method can helps our formal specification by discussing the relation between the domain description, general model and the cases, firstly.

Then, as presented in the thesis, we proposed a methodology for desired property evaluation in Chapter 3 and 4. This implies the possibility for us to make them more executable and visualized in the future applications. Concreted speaking, a piece of software can be developed for evaluating the government web service, according to the methodology we proposed and property we described.

Thirdly, the methodology can be applied for analyzing more different cases, and revising the model we proposed and make the property description more abundant. And to speak more ambitiously, we can include the law making process in our model to achieve even more socialized and culture topics in the future.

Cases Study

By the case studies, we did till now, we can employ other more cases as well as give more concerned and setting down detailed depending on government system development and requirement from people who application domain. We can consider even more improvement methodology than license language up to different situations to achieve more high level of transparent system.

Application

A public license language has been proposed in Chapter 6. The license language can help describe more transparency property, but more strong evidence need to be found to show it real work on transparency improvement for a system. And development of license language interpreter and its application in real government systems is also another possible candidate for our further study.
Appendix A

Glossary

**Accountability** is a concept in ethics with several meanings. It is often used synonymously with such concepts as answerability, enforcement, responsibility, blameworthiness, liability and other terms associated with the expectation of account-giving. As an aspect of governance, it has been central to discussions related to problems in both the public and private (corporation) worlds. Accountability is defined as “A is accountable to B when A is obliged to inform B about As (past or future) actions and decisions, to justify them, and to suffer punishment in the case of eventual misconduct [67] ”.

In leadership roles, accountability is the acknowledgment and assumption of responsibility for actions, products, decisions, and policies including the administration, governance and implementation within the scope of the role or employment position and encompassing the obligation to report, explain and be answerable for resulting consequences.

**Accessability** is a general term used to describe the degree to which a product (e.g., device, service, environment) is accessible by as many people as possible. Accessibility can be viewed as the “ability to access” the functionality, and possible benefit, of some system or entity. Accessibility is often used to focus on people with disabilities and their right of access to entities, often through use of assistive technology. Several definitions of accessibility refer directly to access-based individual rights laws and regulations. Products or services designed to meet these regulations are often termed Easy Access or Accessible. In this thesis, accessibility is the concept which strongly related with transparency. The information accessibility is considered as one of the important target of e-Government system transparency.

**Action** has different meanings in different situations. Generally speaking, it refers to an assertion about the nature of motion. In politics and society, it means something a person can do or a taking into account actions and reactions of other individuals.

**Actor** generally speaking, it refers to a person who acts in a dramatic production and who works in film, television, theater, or radio in that capacity. In this research, actor especially represents the person who is involved into the e-government systems, such as: applicant (citizen), processing officials, administrators and etc.
Agency has multi-meanings depending on context. For instance, in business area, it may refers to a representative, acting on behalf of another. In this thesis, this word mainly means government agency, namely, a department of a local or national government responsible for the oversight and administration of a specific function.

Authorization in security engineering and computer security, authorization is the concept of allowing access to resources only to those permitted to use them. More formally, authorization is a process (often part of the operating system) that protects computer resources by only allowing those resources to be used by resource consumers that have been granted authority to use them. Resources include individual files’ or items’ data, computer programs, computer devices and functionality provided by computer applications. Examples of consumers are computer users, computer programs and other devices on the computer. Authorization (deciding whether to grant access) is a separate concept to authentication (verifying identity), and usually dependent on it.

Behavior refers to a sequence of actions and events [17].

Business License are permits issued by government agencies that allow individuals or companies to conduct business within the government’s geographical jurisdiction. A single jurisdiction often requires multiple licenses that are issued by multiple government departments and agencies. Business licenses vary between countries, states, and local municipalities. There are often many licenses, registrations and certifications required to conduct a business in a single location.

Corruption is essentially termed as an “impairment of integrity, virtue or moral principle; depravity, decay, and/or an inducement to wrong by improper or unlawful means, a departure from the original or from what is pure or correct, and/or an agency or influence that corrupts.”¹

Domain Engineer by domain engineering we mean the engineering of domain descriptions, that is, of their development: (i) from domain capture and analysis (ii) via synthesis, i.e., the domain description document itself, (iii) to its validation with stake-holders and its possible theory development [17].

E-Document refers to electronic document. It is any electronic media content (other than computer programs or system files) that are intended to be used in either an electronic form or as printed output. Originally, any computer data were considered as something internal the final data output was always on paper. However, the development of computer networks has made it so that in most cases it is much more convenient to distribute electronic documents than printed ones. And the improvements in electronic display technologies mean that in most cases it is possible to view documents on screen instead of printing them (thus saving paper and the space required to store the printed copies). However, using electronic documents for final presentation instead of paper has created the problem of multiple incompatible file formats. Even plain text computer files are not free from this problem e.g. under MS-DOS, most programs could not

¹Merriam Webster Dictionary
work correctly with UNIX-style text files (see newline), and for non-English speakers, the different code pages always have been a source of trouble. Even more problems are connected with complex file formats of various word processors, spreadsheets and graphics software. To alleviate the problem, many software companies distribute free file viewers for their proprietary file formats (one example is Adobe’s Acrobat Reader). The other solution is the development of standardized non-proprietary file formats (such as HTML and Open Document), and electronic documents for specialized uses have specialized formats—the specialized electronic articles in physics use TeX or Post Script.

**E-Government** in the thesis, the word refers to government imports information and communication technology for delivering government service.

**E-Governance** here, we are pursuing more than e-government. Governance has been defined as the rules of the political system to solve conflicts between actors and adopt decision (legality). It has also been used to describe the “proper functioning of institutions and their acceptance by the public” (legitimacy). And it has been used to invoke the efficacy of government and the achievement of consensus by democratic means (participation). By applying information and communication technology in government, we are expecting more democracy and transparent governance than before.

**Entity** we loosely understand as something fixed, immobile or static. Although that thing may move, after it has moved it is essentially the same thing, an entity [17].

**Event** refers to the occurrence of something that may either trigger an action, or is triggered by an action, or alter the course of a behaviour, or a combination of these [17].

**Financial Budget** in the thesis refers to the budget of a government, is a summary or plan of the intended revenues and expenditures of that government.

**Function** something, a mathematical quantity (that no one has ever seen), which when applied to something (else), called an argument of the function, yields something (yet else), called a result of the function for that argument. If the function is applied to something which is not a proper argument of the function, then the totally undefined result, called chaos, is yielded [17].

**Formal** the term formal has a number of uses. In the thesis, the word formal particular refers to language or method with mathematically-based or logic-based techniques for the specification, development and verification of software and hardware systems.

**Informal** comparing with the formal in the thesis, means narrative, the description language or method without mathematically-based or logic-based.

**Information** as a concept has a diversity of meanings, from everyday usage to technical settings. Generally speaking, the concept of information is closely related to notions of constraint, communication, control, data, form, instruction, knowledge, meaning, mental stimulus, pattern, perception, and representation.
Knowledge is defined (Oxford English Dictionary) variously as (i) expertise, and skills acquired by a person through experience or education; the theoretical or practical understanding of a subject, (ii) what is known in a particular field or in total; facts and information or (iii) awareness or familiarity gained by experience of a fact or situation. Philosophical debates in general start with Plato’s formulation of knowledge as “justified true belief”. There is however no single agreed definition of knowledge presently, nor any prospect of one, and there remain numerous competing theories. Knowledge acquisition involves complex cognitive processes: perception, learning, communication, association and reasoning. The term knowledge is also used to mean the confident understanding of a subject with the ability to use it for a specific purpose if appropriate. See Knowledge Management for additional details on that discipline.

Message in its most general meaning is an object of communication. It is something which provides information; it can also be this information itself. Therefore, its meaning is dependent upon the context in which it is used; the term may apply to both the information and its form.

New Public Management in the last decade there has been a shift in western countries away from the institutional model that had dominated in the middle and later part of the twentieth century, with the growth of what has come to be known as New Public Management (NPM). The features of NPM are summarised by Hancock (1999: 50) as: Managing public services in the same way as private business; a move from accountability through process to accountability for results; Emphasis on generic management rather than discipline expertise; Devolution of control under strict accounting systems; Separation of core and peripheral functions; Separation of policy-making from the provision of services; Marketisation (including privatisation, contractualism and competitiveness); An emphasis on quantifiable economic definitions of efficiency.

Policy is a deliberate plan of action to guide decisions and achieve rational outcome(s). The term may apply to government, private sector organizations and groups, and individuals. Presidential executive orders, corporate privacy policies, and parliamentary rules of order are all examples of policy. Policy differs from rules or law. While law can compel or prohibit behaviors (e.g. a law requiring the payment of taxes on income) policy merely guides actions toward those that are most likely to achieve a desired outcome.

Policy or policy study may also refer to the process of making important organizational decisions, including the identification of different alternatives such as programs or spending priorities, and choosing among them on the basis of the impact they will have. Policies can be understood as political, management, financial, and administrative mechanisms arranged to reach explicit goals.

Public Administration can be broadly described as the development, implementation and study of branches of government policy. The pursuit of the public good by enhancing civil society and social justice is the ultimate goal of the field. Though public administration has historically referred to as government management, it
increasingly encompasses non-governmental organizations (NGOs) that also operate with a similar, primary dedication to the betterment of humanity.

**Public Agency** see agency.

**Public Government** is same as the word public administration in some place in the thesis. It mainly uses for emphasizing the activities of government branches rather than citizen.

**Public Sector** is the part of economic and administrative life that deals with the delivery of goods and services by and for the government, whether national, regional or local/municipal. Examples of public sector activity range from delivering social security, administering urban planning and organizing national defenses. The organization of the public sector (public ownership) can take several forms, including: (i) Direct administration funded through taxation; the delivering organization generally has no specific requirement to meet commercial success criteria, and production decisions are determined by government. (ii) Publicly owned corporations (in some contexts, especially manufacturing, "state-owned enterprises"); which differ from direct administration in that they have greater commercial freedoms and are expected to operate according to commercial criteria, and production decisions are not generally taken by government (although goals may be set for them by government). (iii) Partial outsourcing (of the scale many businesses do, e.g. for IT services), is considered a public sector model.

**Social Welfare** sometimes is used as synonymously with "human services", as in the USA, to mean income support [5]. This may not be agreeable to everyone in the Australian context, but it avoids the ambiguities that "health" is also a social welfare service. The alternative of "social services" has no foundation across Australia. Also, in Australia "allied health" refers to dietetics, orthotics, occupational therapy, optometry, physiotherapy, podiatry, radiography and speech pathology; in many health services it may include pharmacy, psychology and social work, but this is not found in every instance. Normally nursing is not included in allied health, as it might be in the USA - one notable exception is Battersby et al. In the thesis, for illustration our methodology from formal view, the word social welfare are mainly used to present government service to help low income citizen. And other discussions such as, health care\(^2\) as well.

**Software Engineering** art/discipline/craft/science/logic: Software engineering is the are, discipline, craft, science, logic and practice of synthesizing (i.e., building, constructing) software, i.e., technology, based on scientific insight, and analyzing (i.e., studying, investigating) existing software technology in order to ascertain and discover its possible scientific content.

To succeed in this, Software engineering – abstraction and specification: software engineering makes use of abstraction and specification [16].

**Transparent/Transparency** about the details see Chapter 4. If no special explanation, these two words both refer to government transparency while appearing in the thesis.

\(^2\)can be found from ARIMOTO Yasuhiro's work in LDL lab
Appendix B

OTS/CafeOBJ Specifications of the Domain

--> ***************************************************************
--> Public Administration Domain Description
--> ***************************************************************

mod! SET (P :: TRIV){
  [Elt.P < Set]
  pr(EQL)
  op empty : -> Set
  op add : Elt.P Set -> Set
  op sub : Elt.P Set -> Set
  op _/\in_ : Elt.P Set -> Bool
  --
  op _U_ : Set Set -> Set
  op _&_ : Set Set -> Set
  --
  op subset : Set Set -> Bool
  --
  vars P P1 P2 : Elt.P
  vars S S1 S2 : Set
  --
  eq P1 /\in\ empty = false .
  eq P2 /\in\ add(P1, S) = if (P2 = P1) then true else (P2 /\in\ S) fi .
  eq sub(P1, empty) = empty .
  eq sub(P1, add(P2, S)) = (if (P1 = P2) then sub(P1, S) else add(P2, sub(P1, S)) fi) .
  eq P1 /\in\ (S1 U S2) = (P1 /\in\ S1) or (P1 /\in\ S2) .
  eq P1 /\in\ (S1 & S2) = (P1 /\in\ S1) and (P1 /\in\ S2) .
  --
  eq subset(empty, S) = true .
  eq subset(add(P1, S1), S) = if P1 /\in\ S then subset(S1, S) else false fi .
}

mod! PALIST(X :: TRIV){
  pr(EQL)
  [Nil NList < PList]
  -- basic definition of List
  op nil : -> Nil
  op add : Elt.X PList -> NList
  op car_ : NList -> Elt.X
  op cdr_ : NList -> PList
  var NnL : NList
  var N : Elt.X
  eq (nil = NnL) = false .
  eq car( add( N,nil ) ) = N .
  eq cdr( add( N,nil ) ) = nil .

106
eq car(add(N,NnL)) = N.
eq cdr(add(N,NnL)) = NnL.

-- extend operation on List for PA
op _in_ : Elt.X PAlist -> Bool
op sub : Elt.X PAlist -> PAlist

vars E E' : Elt.X
var L : PAlist

-- define operation to check if E in List
eq E in add(E,nil) = true.
eq E in nil = false.
eq E in add(E',L) = E in L or (E = E')

-- define sub an element from PA list
eq sub(E, add(E,nil)) = nil.
eq sub(E, add(E,NnL)) = NnL.
ceq sub(E,add(E',NnL)) = sub(E,NnL) if not(E = E')

-- Domain Entity & Function
mod* NAME {
    [ActorName DocName < Name]
    pr(EQL)
    op committee : -> ActorName
}

-- Define the possible status of an actor can be
mod! ACTORSTATUS { 
    pr(EQL)
    [ActorStatus]
    ops operating exiting : -> ActorStatus
    eq (operating = exiting) = false.
}

mod! ACTORTYPE { 
    pr(EQL)
    [ActorType]
    ops lawmaker lawinterpreter lawenforcer citizen : -> ActorType
    eq (lawmaker = lawinterpreter) = false.
eq (lawmaker = lawenforcer) = false.
eq (lawmaker = citizen) = false.
eq (lawinterpreter = lawenforcer) = false.
eq (lawinterpreter = citizen) = false.
eq (lawenforcer = citizen) = false.
}

-- Define Operation in Public Administration Domain
mod! OPERATION { 
    pr(EQL)
    [Operation]
    ops creat edit read copy calculate shred auth : -> Operation
    eq (creat = edit) = false.
eq (creat = read) = false.
eq (creat = copy) = false.
eq (creat = calculate) = false.
eq (creat = shred) = false .
eq (edit = read) = false .
eq (edit = copy) = false .
eq (edit = calculate) = false .
eq (edit = shred) = false .
eq (read = copy) = false .
eq (read = calculate) = false .
eq (read = shred) = false .
eq (copy = calculate) = false .
eq (copy = shred) = false .
eq (calculate = shred) = false .
eq (auth = creat) = false .
eq (auth = edit) = false .
eq (auth = read) = false .
eq (auth = calculate) = false .
eq (auth = copy) = false .
eq (auth = shred) = false .

mod! PAIR(M :: TRIV, N :: TRIV) {
  pr(EQL)
  [Pair]
  op [_,_] : Elt.M Elt.N -> Pair
  op left : Pair -> Elt.M
  op right : Pair -> Elt.N

  var P : Pair
  vars X1 X2 : Elt.M
  vars Y1 Y2 : Elt.N

  eq left([X1, Y1]) = X1 .
  eq right([X1, Y1]) = Y1 .
  eq ([X1, Y1] = [X2, Y2]) = (X1 = X2) and (Y1 = Y2) .
}

mod! DOCOPERATION {
  pr (PAIR (M <= view to NAME sort Elt -> DocName),
       N <= view to OPERATION sort Elt -> Operation))
}

mod! ACTOROPERATION {
  pr (PAIR (M <= view to NAME sort Elt -> ActorName,
            N <= view to OPERATION sort Elt -> Operation))
}

mod* OPERATIONSET {
  pr (SET(P <= view to DOCOPERATION sort Elt -> DocOperation)
      * sort Set -> OperationSet ,
      op empty -> em-os )
}

mod* OPERATIONLIST {
  pr (PALIST(X <= view to ACTOROPERATION sort Elt -> ActorOperation)
      * sort PAlist -> OperationList ,
      op nil -> nil-os )
}

mod! TPAIR( M :: TRIV, N :: TRIV, O :: TRIV ) {
  pr(EQL)
  [TPair]
  op < ..., ... > : Elt.M Elt.N Elt.O -> TPair
  op hed : TPair -> Elt.M
  op mid : TPair -> Elt.N
  op lst : TPair -> Elt.O

108
var P : TPair
vars X1 X2 : Elt.M
vars Y1 Y2 : Elt.N
vars Z1 Z2 : Elt.O
eq hed( < X1, Y1, Z1 > ) = X1 .
eq mid( < X1, Y1, Z1 > ) = Y1 .
eq lst( < X1, Y1, Z1 > ) = Z1 .
eq ( < X1, Y1, Z1 > = < X2, Y2, Z2 > ) = ( X1 = X2 ) and ( Y1 = Y2 ) and ( Z1 = Z2 ) .

mod! DOCAUTHORIZATION
pr (TPAIR (M <= view to NAME
{ sort Elt -> DocName ,
N <= view to OPERATION sort Elt -> Operation ,
O <= view to NAME
{ sort Elt -> ActorName })
*{sort TPair -> DocAuthorization})

mod* AUTHORIZATIONSET
pr (SET(P <= view to DOCAUTHORIZATION { sort Elt -> DocAuthorization })
* { sort Set -> AuthorizationSet ,
op empty -> em-au })

mod* AUTHORIZATIONLIST
pr (PALIST(X <= view to DOCAUTHORIZATION {sort Elt -> DocAuthorization})
* {sort PAlist -> AuthorizationList ,
op nil -> nil-au})

-- Define Actor in public administration domain
mod! ACTOR
pr(EQL + NAME + ACTORSTATUS + ACTORTYPE + AUTHORIZATIONSET + OPERATIONSET )
[Actor]
op mk-actor : ActorName ActorStatus ActorType OperationSet AuthorizationSet -> Actor
op.actorname : Actor -> ActorName
op actortype : Actor -> ActorType
op operationrecord : Actor -> OperationSet
op.authorizationset : Actor -> AuthorizationSet

--
var AN : ActorName
var AS : ActorStatus
var AT : ActorType
var OS : OperationSet
var AUS : AuthorizationSet

--
eq actorname(mk-actor(AN, AS, AT, OS, AUS )) = AN .
eq actorstatus(mk-actor(AN, AS, AT, OS, AUS)) = AS .
eq actortype(mk-actor(AN, AS, AT, OS, AUS)) = AT .
eq operationrecord(mk-actor(AN, AS, AT, OS, AUS)) = OS .
eq authorizationset(mk-actor(AN, AS, AT, OS, AUS)) = AUS .

-- Define Document STATUS
mod! DOCUMENTSTATUS
pr(EQL)
[DocumentStatus]
ops creat edited readed copied calculated idled shreded auth : -> DocumentStatus

eq (edited = readed) = false .
eq (edited = copied) = false .
eq (edited = calculated) = false .
eq (edited = idled) = false .
eq (edited = shreded) = false .
eq (readed = copied) = false .
eq (readed = calculated) = false .
eq (readed = idled ) = false .
eq (readed = shreded) = false .
eq (copied = calculated) = false .
eq (copied = idled ) = false .
eq (copied = shreded) = false .
eq (calculated = idled ) = false .
eq (calculated = shreded) = false .
eq (idled = shreded) = false .

mod! DOCUMENTTYPE {
pr(EQL)
[DocumentType]
ops lmdoc lidoc ledoc citizendoc : -> DocumentType
eq (lmdoc = lidoc) = false .
eq (lmdoc = ledoc) = false .
eq (lmdoc = citizendoc) = false .
eq (lidoc = ledoc) = false .
eq (lidoc = citizendoc) = false .
eq (ledoc = citizendoc) = false .
}

mod! INFORMATION {
pr(EQL)
[Information]
op commonsense : -> Information
}

mod! DOCUMENT {
pr(EQL + DOCUMENTSTATUS + NAME + DOCUMENTTYPE + INFORMATION + AUTHORIZATIONLIST + OPERATIONLIST)
[Document]
op mk-document : DocName ActorName DocumentStatus DocumentType Information OperationList AuthorizationList -> Document
op _=_ : Document Document -> Bool

--
op docname : Document -> DocName
op doccreator : Document -> ActorName
op docstate : Document -> DocumentStatus
op doctype : Document -> DocumentType
op docinfo : Document -> Information
op doctrace : Document -> OperationList
op authrecord : Document -> AuthorizationList

--
var DN : DocName
var DC : ActorName
var DS : DocumentStatus
var DT : DocumentType
var DI : Information
var OL : OperationList
var AH : AuthorizationList

--
eq (docname(mk-document(DN, DC, DS, DT, DI, OL, AH )) = DN .
eq (doccreator(mk-document(DN, DC, DS, DT, DI, OL, AH )) = DC .
eq (docstate(mk-document(DN, DC, DS, DT, DI, OL, AH )) = DS .
eq (doctype(mk-document(DN, DC, DS, DT, DI, OL, AH )) = DT .
eq (docinfo(mk-document(DN, DC, DS, DT, DI, OL, AH )) = DI .
eq (doctrace(mk-document(DN, DC, DS, DT, DI, OL, AH )) = OL .
eq (authrecord(mk-document(DN, DC, DS, DT, DI, OL, AH )) = AH .
}

--> **************************************************************
--> Public Administration Domain System
--> **************************************************************

mod* PA {
pr (ACTOR + DOCUMENT + OPERATION)
pr (EQL)
*[System]*

-- any initial state
op init : -> System

-- observations in PA System
bop actor : System ActorName -> Actor
bop document : System DocName -> Document

-- actions
bop authorize : System ActorName ActorName DocName Operation -> System
bop operate : System ActorName DocName Operation -> System

-- variable in CafeOBJ
var S : System
vars A1 A2 A : ActorName
vars D1 D2 D : DocName
vars O1 O2 O : Operation

-- initial state
eq actor(init, A) = mk-actor(A, exiting, citizen, em-os, em-au) .
eq document(init, D) = mk-document(D, committee, idled, 1mdoc, commonsense, nil-os, nil-au) .

-- authorize
op c-authorize : System ActorName ActorName DocName Operation -> Bool
eq c-authorize(S, A1, A2, D, O) =
   actorstatus(actor(S, A1)) = exiting and actorstatus(actor(S, A2)) = exiting
   and ((A1 = doccreator(document(S,D))) or (D, O, A1) /in authorizationset(actor(S, A1))) .

--
ceq actor(authorize(S, A1, A2, D, O), A1) =
   (if (not(A1 = A) and not(A2 = A)) then actor(S, A)
   else if(A = A2) then mk-actor(A2, exiting, actortype(actor(S, A2)),
   operationrecord(actor(S, A2)), add(< D, O, A2 >, authorizationset(actor(S, A2))))
   else mk-actor(A1, exiting, actortype(actor(S, A1)), operationrecord(actor(S, A1)),
   authorizationset(actor(S, A1))) fi)

ceq document(authorize(S, A1, A2, D, O), D1) =
   (if not(D = D1) then document(S, D)
   else mk-document(D, doccreator(document(S,D)), idled, doctype(document(S,D)),
   docinfo(document(S,D)), doctrace(document(S,D)),
   add(< D, O, A1 >, authrecord(document(S,D)))) fi)

ceq authorize(S, A1, A2, D, O) = S if not c-authorize(S, A1, A2, D, O) .

-- operate
op c-operate : System ActorName DocName Operation -> Bool
eq c-operate(S, A, D, O) =
   < D, O, A > /in authorizationset(actor(S, A)) and (docstate(document(S,D)) = idled) or (O = creat) .

--
ceq actor(operate(S, A, D, O), A1) =
   (if not(A1 = A) then actor(S, A)
   else mk-actor(A1, operating, actortype(actor(S, A1)), add([D, O],
   operationrecord(actor(S, A1))), authorizationset(actor(S, A1))) fi)

ceq document(operate(S, A, D, O), D1) =
   (if not(D = D1) then document(S, D)
   else mk-document(D, doccreator(document(S,D)), idled, doctype(document(S,D)),
   docinfo(document(S,D)), add([A, O], doctrace(document(S,D))),
   authrecord(document(S,D)))) fi)

ceq operate(S, A, D, O) = S if not c-operate(S, A, D, O) .
}
Appendix C

General Service Delivery (GSD) Model

C.1 Specification of GSD Model

-- CafeOBJ specification for a general workflow modules
-- in Public Administration System.

--> *************************
--> Part I: General modules
-->*************************

mod! SET (P :: TRIV) {
  [Elt.P < Set]
  pr(EQL)
  op empty : -> Set
  op add : Elt.P Set -> Set
  op sub_ : Elt.P Set -> Set
  op_/in_ : Elt.P Set -> Bool
  --
  op _U_ : Set Set -> Set
  op _&_ : Set Set -> Set
  --
  op subset : Set Set -> Bool
  --
  vars P P1 P2 : Elt.P
  vars S S1 S2 : Set
  --
  eq P1 /in empty = false .
  eq P2 /in add(P1, S) = if (P2 = P1) then true else (P2 /in S) fi .
  eq sub(P1, empty) = empty .
  eq sub(P1, add(P2, S)) = (if (P1 = P2) then sub(P1, S) else add(P2, sub(P1, S)) fi) .
  eq P1 /in (S1 U S2) = (P1 /in S1) or (P1 /in S2) .
  eq P1 /in (S1 & S2) = (P1 /in S1) and (P1 /in S2) .
  --
  eq subset(empty, S) = true .
  eq subset(add(P1, S1), S) = if P1 /in S then subset(S1, S) else false fi .
}

mod! LIST(X :: TRIV) {
  pr(EQL)
  [Nil Elt.X < NList < List]
  -- basic definition of List
  op nil : -> Nil
  op add : Elt.X List -> NList
  op car_ : NList -> Elt.X
  op cdr_ : NList -> List
  var NaL : NList

112
var N : Elt.X

eq (nil = Nnil) = false .
eq car( add( N,nil ) ) = N .
eq cdr( add( N,nil ) ) = nil .
eq car( add( N,NnL ) ) = N .
eq cdr( add( N,NnL ) ) = NnL .

-- extend operation on List for Document
op _/ in_ : Elt.X List -> Bool
op sub : Elt.X List -> List

vars E E' : Elt.X
var L : List

-- define operation to check if E in List
eq E / in nil = false .
eq E / in add(E,nil) = true .
eq E / in add(E',L) = E / in L or (E = E') .

-- define sub an element from list
eq sub(E, nil) = nil .
eq sub(E, add(E,nil)) = nil .
eq sub(E,add(E',NnL)) = (if (E = E') then sub(E,NnL) else add(E',sub(E,NnL)) fi) .
}

mod! PAIR(M :: TRIV, N :: TRIV) {
pr(EQL)
[Pair]
--
op [_,_] : Elt.M Elt.N -> Pair
op left : Pair -> Elt.M
op right : Pair -> Elt.N

var P : Pair
vars X1 X2 : Elt.M
vars Y1 Y2 : Elt.N
eq left([X1, Y1]) = X1 .
eq right([X1, Y1]) = Y1 .
eq ([X1, Y1] = [X2, Y2]) = (X1 = X2) and (Y1 = Y2) .
}

mod! TRI(M :: TRIV, N :: TRIV, O :: TRIV) {
pr(EQL)
[Tri]
--
op left : Tri -> Elt.M
op mid : Tri -> Elt.N
op right : Tri -> Elt.O
op _=_ : Tri Tri -> Bool

var T : Tri
vars X1 X2 : Elt.M
vars Y1 Y2 : Elt.N
vars Z1 Z2 : Elt.O
eq left([X1, Y1, Z1]) = X1 .
eq mid([X1, Y1, Z1]) = Y1 .
eq right([X1, Y1, Z1]) = Z1 .
eq ([X1, Y1, Z1] = [X2, Y2, Z2]) = (X1 = X2) and (Y1 = Y2) and (Z1 = Z2) .
}

mod* NAME {
[ActorName DocName < Name]
pr(EQL)
}

mod! RESULT {
pr(EQL)
[Result]
ops pass unpass : -> Result
--
eq (pass = unpass) = false.
}

--> *****************************
--> Part II : modules for Actors
--> *****************************

mod! ACTORSTATUS {
  [ActorStatus]
  pr(EQL)
  ops handling idling : -> ActorStatus
  --
eq (handling = idling) = false.
}

mod! DOCNAMESET {
  pr (SET(P <= view to NAME sort Elt -> DocName)
      *sort Set -> DocNameSet)
  pr(EQL)
  [DocNameSetConst < DocNameSet]
  --
op emptyDocNameSet : -> DocNameSet
  op basicLawSet : -> DocNameSet
  ops basicCitiLawSet basicEnLawSet basicInLawSet basicMakerLawSet : -> DocNameSet
  --
  vars DNS1 DNS2 : DocNameSetConst
  eq (DNS1 = DNS2) = (DNS1 == DNS2).
}

mod! ACTORROLESET {
  pr (SET(P <= view to NAME sort Elt -> ActorName)
      *sort Set -> ActorRoleSet)
  --
  -- three roles of the public administration system
  ops applicantSet agenciesSet : -> ActorRoleSet
  --
op _=_ : ActorRoleSet ActorRoleSet -> Bool comm
  var ARS : ActorRoleSet
  eq (ARS = ARS) = true.
  --
eq (applicantSet = agenciesSet) = false.
}

mod! OPERATION {
  [OpConst < Operation]
  pr(EQL)
  ops apply evaluate confirm : -> OpConst
  ops notify execute track : -> OpConst
  --
  vars Op1 Op2 : OpConst
  eq (Op1 = Op2) = (Op1 == Op2).
}

mod! AUTHORIZATION {
  [Mand Auth < Authorization]
  pr(EQL + OPERATION + ACTORROLESET)
  op auth : Name Operation ActorRoleSet -> Auth
  op mand : Name Operation ActorRoleSet -> Mand
  --
  op nameAuth : Authorization -> Name
  op opAuth : Authorization -> Operation
  op nameSetAuth : Authorization -> ActorRoleSet
-- vars N N1 N2 : Name
vars O O1 O2 : Operation
vars A A1 A2 : ActorRoleSet
vars AU1 AU2 : Auth
vars M1 M2 : Mand

--
eq nameAuth(auth(N,O,A)) = N .
eq opAuth(auth(N,O,A)) = O .
eq nameSetAuth(auth(N,O,A)) = A .
--
eq nameAuth(mand(N,O,A)) = N .
eq opAuth(mand(N,O,A)) = O .
eq nameSetAuth(mand(N,O,A)) = A .
--
eq (AU1 = AU2) = (nameAuth(AU1) = nameAuth(AU2) and opAuth(AU1) = opAuth(AU2) and 
nameSetAuth(AU1) = nameSetAuth(AU2)) .
eq (M1 = M2) = (nameAuth(M1) = nameAuth(M2) and opAuth(M1) = opAuth(M2) and 
nameSetAuth(M1) = nameSetAuth(M2)) .
eq (auth(N1,O1,A1) = mand(N2,O2,A2)) = false .
}

mod! OPSET {
  pr (SET(P <= view to OPERATION { sort Elt -> Operation }))
    *(sort Set -> OSet))
}

mod! AUTHSET {
  pr (SET (P <= view to AUTHORIZATION sort Elt -> Authorization )
    *sort Set -> AuthSet)
}

mod! DOCSTATUS {
  pr(EQL)
  [DocStatus]
  --
  ops operating exiting : -> DocStatus
  --
  eq (operating = exiting) = false .
}

mod! DOCPAIR {
  pr (PAIR(M <= view to NAME {sort Elt -> ActorName},
    N <= view to OPERATION {sort Elt -> Operation }))
    *(sort Pair -> DocPair )
}

-- Document Trace can be viewed as a list
-- to record the document revision history
-- Two components:
-- (1) actor who operates on the document
-- (2) operation on the document.
mod! DOCTRACE {
  pr(LIST(X <= view to DOCPAIR { sort Elt -> DocPair}))
    *(sort List -> DocTrace ,
    op nil => ndt )
  pr(OPERATION)
}

op emptyDocPair : -> DocPair
op getTerm : Operation DocTrace -> DocPair
--
var O : Operation
var A : ActorName
var DP : DocPair
mod! DOCLABEL {
  pr(EQL)
  [DocLabelConst < DocLabel]
  --
  ops applied evaluated confirmed : -> DocLabel
  ops notified executed issued : -> DocLabel
  op emptied : -> DocLabel
  --
  vars DLc1 DLc2 : DocLabelConst
  eq (DLc1 = DLc2) = (DLc1 == DLc2) .
  eq (applied = evaluated) = false .
  eq (applied = confirmed) = false .
  eq (applied = notified) = false .
  eq (applied = executed) = false .
  eq (applied = issued) = false .
  eq (applied = emptied) = false .
  eq (evaluated = confirmed) = false .
  eq (evaluated = notified) = false .
  eq (evaluated = executed) = false .
  eq (evaluated = issued) = false .
  eq (evaluated = emptied) = false .
  eq (confirmed = notified) = false .
  eq (confirmed = executed) = false .
  eq (confirmed = issued) = false .
  eq (confirmed = emptied) = false .
  eq (notified = executed) = false .
  eq (notified = issued) = false .
  eq (notified = emptied) = false .
  eq (executed = issued) = false .
  eq (executed = emptied) = false .
}

mod! ACTOR {
  pr(EQL + NAME + ACTORSTATUS + DOCNAMESET + DOCLABEL)
  [Actor]

  op mk-actor : ActorName ActorStatus DocNameSet DocLabel -> Actor
  op actorName : Actor -> ActorName
  op actorStatus : Actor -> ActorStatus
  op actorKnow : Actor -> DocNameSet
  op actorTrack : Actor -> DocLabel
  op _=_ : Actor Actor -> Bool comm
  --
  vars A1 A2 : Actor
  var N : ActorName
  var S : ActorStatus
  var DS : DocNameSet
  var DL : DocLabel
  --
  eq actorName(mk-actor(N,S,DS,DL)) = N .
  eq actorStatus(mk-actor(N,S,DS,DL)) = S .
  eq actorKnow(mk-actor(N,S,DS,DL)) = DS .
  eq actorTrack(mk-actor(N,S,DS,DL)) = DL .
  --
  eq (A1 = A2) = actorName(A1) = actorName(A2) .
}

---> ***********************************
--->Part III : modules for Documents
--->******************************************************************************
mod! DOCAUTH 
pr (LIST(X <= view to AUTHORIZATION {sort Elt -> Authorization}))
    * { sort List -> DocAuth ,
          op nil -> nda } )

    op del : Authorization DocAuth -> DocAuth
--
    vars A1 A2 : Authorization
    var DA : DocAuth
--
    eq del(A1,nda) = nda .
    eq del(A1,add(A2,DA)) = (if (A1 = A2) then del(A1,DA) else add(A2,del(A1,DA)) fi) .

mod! DOCTYPE 
pr(EQL)
[DocType]
--
    ops pdoc ddoc ldoc : -> DocType
--
    eq (pdoc = ddoc) = false .
    eq (pdoc = ldoc) = false .
    eq (ddoc = ldoc) = false .

mod! INFORMATION 
[Info]
pr(EQL)
ops none general noenough : -> Info
--
    op _=_ : Info Info -> Bool comm
    op _@_ : Info Info -> Info
--
    vars I I1 I2 : Info
--
    eq (none = general) = false .
    eq (I = I) = true .
--
    eq (none @ none) = none .
    eq (none @ I) = I .
--
    eq (none < I) = true .
    eq (I < general) = true .
    eq (I1 @ I2) = if I1 < I2 then I2 else I1 fi .
--

mod! DOCUMENT 
[Application < Doc]
pr(EQL + NAME + DOCSTATUS + DOCTYPE + INFORMATION + DOCNAMESET + DOCTRACE + DOCAUTH + DOCLABEL)

    op mk-doc : DocName ActorName DocStatus DocType Info
                DocTrace DocAuth DocNameSet DocLabel -> Doc
    op _=_ : Doc Doc -> Bool comm
--
    -- constant "emptyDoc" is used to denote the initial value of a doc that is not created yet.
    op emptyDoc : -> Doc
    -- constant "basicLaw" is used to denote the initial value of a doc in basicLawSet
    op basicLaw : -> Doc
op docName : Doc -> DocName
op docCreator : Doc -> ActorName
op docStatus : Doc -> DocStatus
op docType : Doc -> DocType
op docInfo : Doc -> Info
op docTrace : Doc -> DocTrace
op docAuth : Doc -> DocAuth
op docRefer : Doc -> DocNameSet
op docLabel : Doc -> DocLabel

vars D D1 D2 : Doc
vars N N1 N2 : DocName
var A : ActorName
var S : DocStatus
var T : DocType
var I : Info
var DT : DocTrace
var DA : DocAuth
var DS : DocNameSet
var DL : DocLabel

-- op bl : -> DocName .
--
eq (docName(basicLaw) /in basicLawSet) = true .
eq (docCreator(basicLaw) = A) = false .
eq docTrace(basicLaw) = ndt .
eq docLabel(basicLaw) = issused .
eq docInfo(basicLaw) = general .
eq docLabel(emptyDoc) = emptied .
eq (docName(emptyDoc) = N) = false .
eq (docCreator(emptyDoc) = A) = false .
eq docTrace(emptyDoc) = ndt .

--** added part -- eq (docInfo(D) < docInfo(basicLaw)) = true .
eq (docInfo(emptyDoc) < docInfo(D)) = true .
--** added end

--
eq (D1 = D2) = (docName(D1) = docName(D2)) .
-- the following function checks if there exists at least one document whose label is issused
op hasiss : DocNameSet -> Bool

-- initial state
op init : -> Sys

-- observer
bop actor : Sys ActorName -> Actor
bop doc : Sys DocName -> Doc

-- actions

-- part IV : GSD Model
-- *********** Part IV : GSD Model

mod* PA {
  pr (ACTOR + DOCUMENT + AUTHSET + OPSET + ACTORROLESET + RESULT)
  pr (EQL)
  *[Sys]*

  -- initial state
  op init : -> Sys

  -- observer
  bop actor : Sys ActorName -> Actor
  bop doc : Sys DocName -> Doc

  -- actions

  --** added part -- eq (docInfo(D) < docInfo(basicLaw)) = true .
eq (docInfo(emptyDoc) < docInfo(D)) = true .
--** added end

-- eq docName(mk-doc(N,A,S,T,I,DT,DA,DS,DL)) = N .
eq docCreator(mk-doc(N,A,S,T,I,DT,DA,DS,DL)) = A .
eq docType(mk-doc(N,A,S,T,I,DT,DA,DS,DL)) = T .
eq docInfo(mk-doc(N,A,S,T,I,DT,DA,DS,DL)) = I .
eq docTrace(mk-doc(N,A,S,T,I,DT,DA,DS,DL)) = DT .
eq docRefer(mk-doc(N,A,S,T,I,DT,DA,DS,DL)) = DS .

  -- eq (D1 = D2) = (docName(D1) = docName(D2)) .

  -- the following function checks if there exists at least one document whose label is issused
  op hasiss : DocNameSet -> Bool
  eq hasiss(emptyDocNameSet) = false .
eq hasiss(docName(D)) = (if docLabel(D) = issused then true else false fi) .
eq hasiss(add(docName(D), DS)) = (if docLabel(D) = issused then true else hasiss(DS) fi) .
}
bop apply : Sys ActorName DocName Info -> Sys
bop evaluate : Sys ActorName DocName DocNameSet Info -> Sys
bop confirm : Sys ActorName DocName Info -> Sys
bop notify : Sys ActorName DocName Result -> Sys
bop execute : Sys ActorName DocName -> Sys
bop track : Sys ActorName DocName -> Sys

-- variable
var S : Sys
vars C C1 C2 : ActorName
vars D D1 D2 : DocName
vars DS DS1 DS2 : DocNameSet
var I : Info
var R : Result

--> added operation
op sumInfo : DocNameSet -> Info
  eq sumInfo(add(D,emptyDocNameSet)) = docInfo(doc(S,D)) .
  eq sumInfo(add(D,DS)) = (docInfo(doc(S,D)) @ sumInfo(DS)) .

--> *** added end

-- [0] equations of initial state
eq actor(init,C) = mk-actor(C,idling,basicLawSet,issused) .
eq doc(init,D) = (if D /in basicLawSet then basicLaw else emptyDoc fi) .

-- [1] apply
op c-apply : Sys ActorName DocName Info -> Bool
  eq c-apply(S,C,D,I) = doc(S,D) = emptyDoc and C /in applicantSet and not (D /in basicLawSet) .
  --
  ceq actor(apply(S,C,D,I),C1) = (if not(C1 = C) then actor(S,C1)
  else mk-actor(C,handling,actorKnow(actor(S,C)),actorTrack(actor(S,C))) fi)
  if c-apply(S,C,D,I) .
  ceq doc(apply(S,C,D,I),D1) = (if not(D1 = D)
  then doc(S,D1)
  else mk-doc(D,operating,pdoc,I,
  add([C,apply],ndt),
  docAuth(doc(S,D)),
  actorKnow(actor(S,C)),applied) fi)
  if c-apply(S,C,D,I) .
  ceq apply(S,C,D,I) = S if not c-apply(S,C,D,I) .

-- [2] evaluate
op c-evaluate : Sys ActorName DocName DocNameSet Info -> Bool
  eq c-evaluate(S,C,D,DS,I) = C /in agenciesSet and not (C = docCreator(doc(S,D))) and
  docType(doc(S,D)) = pdoc and docLabel(doc(S,D)) = applied
  and (auth(docCreator(doc(S,D)),evaluate,agenciesSet) /in docAuth(doc(S,D))) .
  --
  ceq actor(evaluate(S,C,D,DS,I),C1) = (if not(C1 = C) then actor(S,C1)
  else mk-actor(C,handling,actorKnow(actor(S,C)),actorTrack(actor(S,C))) fi)
  if c-evaluate(S,C,D,DS,I) .
  ceq doc(evaluate(S,C,D,DS,I),D1) = (if not(D1 = D)
  then doc(S,D1)
  else mk-doc(D,docCreator(doc(S,D)),docStatus(doc(S,D)),docType(doc(S,D)),
  (I @ docInfo(doc(S,D))),add([C,evaluate],docTrace(doc(S,D))),
  add(auth(C,confirm,agenciesSet),docAuth(doc(S,D))),
  DS U docRefer(doc(S,D)),evaluated) fi)
  if c-evaluate(S,C,D,DS,I) .
  ceq evaluate(S,C,D,DS,I) = S if not c-evaluate(S,C,D,DS,I) .

119
-- [3] confirm
op c-confirm : Sys ActorName DocName Info -> Bool
eq c-confirm(S,C,D,I)
= C /in agenciesSet and
not (C = docCreator(doc(S,D))) and not (C = left(getTerm(evaluate,docTrace(doc(S,D))))) and
docType(doc(S,D)) = pdoc and docLabel(doc(S,D)) = evaluated and
(auth(left(getTerm(evaluate,docTrace(doc(S,D)))),confirm,agenciesSet) /in docAuth(doc(S,D))) .

--
ceq actor(c-confirm(S,C,D,I),C1)
= (if not(C1 = C) then actor(S,C1)
  else mk-actor(C1,handling,actorKnow(actor(S,C1)), actorTrack(actor(S,C1))) fi)
if c-confirm(S,C,D,I) .

ceq doc(c-confirm(S,C,D,I),D1)
= (if not(D1 = D)
  then doc(S,D1)
  else mk-doc(D,docCreator(doc(S,D)),docStatus(doc(S,D)),docType(doc(S,D)),
    (I @ docInfo(doc(S,D))),add([C,confirm],docTrace(doc(S,D))),
    del(auth(left(getTerm(evaluate,docTrace(doc(S,D)))),evaluate,agenciesSet),
      (if I = noenough then docAuth(doc(S,D))
        else add(auth(C,notify,agenciesSet),docAuth(doc(S,D))) fi)),
      (actorKnow(actor(S,C)) U docRefer(doc(S,D))),
      (if I = noenough then applied else confirmed fi)) fi)
if c-confirm(S,C,D,I) .

ceq confirm(S,C,D,I) = S if not c-confirm(S,C,D,I) .

-- [4] notify
op c-notify : Sys ActorName DocName Result -> Bool
eq c-notify(S,C,D,R)
= C /in agenciesSet and not (C = docCreator(doc(S,D))) and
not (C = left(getTerm(confirm,docTrace(doc(S,D))))) and
docLabel(doc(S,D)) = confirmed and
auth(left(getTerm(confirm,docTrace(doc(S,D)))),notify,agenciesSet) /in docAuth(doc(S,D)) .

--
ceq actor(c-notify(S,C,D,R),C1)
= (if not(C1 = C) then (if C1 = docCreator(doc(S,D))
  then mk-actor(C1,idling,add(D,actorKnow(actor(S,C1))),actorTrack(actor(S,C1)))
  else actor(S,C1) fi)
else mk-actor(C,handling,actorKnow(actor(S,C)), actorTrack(actor(S,C))) fi)
if c-notify(S,C,D,R) .

ceq doc(c-notify(S,C,D,R),D1)
= (if not(D1 = D)
  then doc(S,D1)
  else mk-doc(D,docCreator(doc(S,D)),docStatus(doc(S,D)),ddoc,
    docInfo(doc(S,D))),add([C,confirm],docTrace(doc(S,D))),
    del(auth(left(getTerm(confirm,docTrace(doc(S,D)))),notify,agenciesSet),
      (if R = unpass then docAuth(doc(S,D))
        else add(auth(C,execute,agenciesSet),docAuth(doc(S,D))) fi)),
      docRefer(doc(S,D)),notified) fi)
if c-notify(S,C,D,R) .

ceq notify(S,C,D,R) = S if not c-notify(S,C,D,R) .

-- [5] execute
op c-execute : Sys ActorName DocName -> Bool
eq c-execute(S,C,D)
= C /in agenciesSet and not (C = docCreator(doc(S,D))) and
not (C = left(getTerm(confirm,docTrace(doc(S,D))))) and
docType(doc(S,D)) = ddoc and docLabel(doc(S,D)) = notified and
auth(left(getTerm(confirm,docTrace(doc(S,D)))),execute,agenciesSet) /in docAuth(doc(S,D)) .

--
ceq actor(c-execute(S,C,D),C)
= (if not(C1 = C) then actor(S,C)
  else mk-actor(C1,handling,actorKnow(actor(S,C1)), actorTrack(actor(S,C1))) fi)
if c-execute(S,C,D) .

ceq doc(c-execute(S,C,D),D1)
= (if not(D1 = D)
  then doc(S,D1)
  else mk-doc(D,docCreator(doc(S,D)),docStatus(doc(S,D)),
    docInfo(doc(S,D))),add([C,confirm],docTrace(doc(S,D))),
    del(auth(left(getTerm(confirm,docTrace(doc(S,D)))),execute,agenciesSet),
      (if R = unpass then docAuth(doc(S,D))
        else add(auth(C,execute,agenciesSet),docAuth(doc(S,D))) fi)),
      docRefer(doc(S,D)),notified) fi)
if c-execute(S,C,D) .

ceq execute(S,C,D) = S if not c-execute(S,C,D) .

120
then doc(S,D)
else mk-doc(D,docCreator(doc(S,D)),exiting,docType(doc(S,D)),
docInfo(doc(S,D)),add([C,execute],docTrace(doc(S,D))),
del(auth(left(getTerm(notify,docTrace(doc(S,D)))),execute,agenciesSet),
docAuth(doc(S,D)),docRefer(doc(S,D)),executed) fi)

if c-execute(S,C,D) .

ceq execute(S,C,D) = S if not c-execute(S,C,D) .

-- [6] track
op c-track : Sys ActorName DocName -> Bool
eq c-track(S,C,D) = ((C = docCreator(doc(S,D))) and not (doc(S,D) = emptyDoc)) or (D /in basicLawSet) .

--

cq actor(track(S,C,D),C1)
= (if not(C1 = C) then actor(S,C1)
else mk-actor(C1,handling,actorKnow(actor(S,C1)), docLabel(doc(S,D))) fi)
if c-track(S,C,D) .

cq doc(track(S,C,D),D1)
= (if not (D = D1) then doc(S,D1)
else mk-doc(D1, C, docStatus(doc(S,D1)),docType(doc(S,D1)),
docInfo(doc(S,D1)),add([C,track], docTrace(doc(S,D1))),docAuth(doc(S,D1)),
docRefer(doc(S,D1)),docLabel(doc(S,D1))) fi)
if c-track(S,C,D) .

cq track(S,C,D) = S if not c-track(S,C,D) .
}

-- 1. trivial property of transparency: creator of the document can track his/her application any time

open PA
op a : -> ActorName .
op d : -> DocName .
op s : -> Sys .

red (docCreator(doc(s,d)) = a) implies c-track(s,a,d) .
red docLabel(basicLaw) = issued and c-track(s,a,docName(basicLaw)) .

close

C.2 Transparency Definition based on the GSD Model

mod INV {
  pr(PA)
  -- operators denoting invariants --
  ops a1 a2 a : -> ActorName
  op d : -> DocName
  op r : -> Result
  -- anti-corruption
  op inv1 : Sys ActorName DocName -> Bool
  op inv2 : Sys ActorName DocName -> Bool
  op inv3 : Sys ActorName ActorName DocName -> Bool
  op inv4 : Sys ActorName DocName -> Bool
  op inv5 : Sys ActorName -> Bool
  op inv6 : Sys ActorName DocName -> Bool
  op inv7 : Sys ActorName DocName -> Bool
  op inv8 : Sys ActorName ActorName DocName -> Bool
  -- CafeOBJ variables
  var S : Sys
  vars A1 A2 A : ActorName
  var D : DocName
  -- invariants -- these three wrote by anti-corruption
  eq inv1(S,A,D) =
    [A,evaluate] /in docTrace(doc(S,D)) or [A,confirm] /in docTrace(doc(S,D))
    implies not([A,apply] /in docTrace(doc(S,D))) .
  eq inv2(S,A,D) =
eq inv3(S,A1,A2,D) =  
[A1,confirm] /in docTrace(doc(S,D)) and [A2,execute] /in docTrace(doc(S,D)) implies not(A1 = A2) .
-- these wrote by direct definition of transparency -- actor can track -- eq inv4(S,A,D) =  
c-track(S,A,D) and not(D /in basicLawSet) implies [A,applied] /in docTrace(doc(S,D)) .
-- information about policy know before execute -- eq inv5(S,A) =  
subset(basicLawSet,actorKnow(actor(S,A))) .
-- decision made with reasons -- eq inv6(S,A,D) =  
[A,submitAD] /in docTrace(doc(S,D)) and docType(doc(S,D)) = ddoc implies subset(docRefer(doc(S,D)),actorKnow(actor(S,A))) .
-- accountability
eq inv7(S,A,D) =  
docLabel(doc(S,D)) = confirmed implies [A,confirm] /in docTrace(doc(S,D)) .
-- information integrity -- eq inv8(S,A,D) =  
([A,submitAD] /in docTrace(doc(S,D))) implies (docInfo(doc(S,D)) < sumInfo(actorKnow(actor(S,A)))) .

op lemma1 : Sys ActorName DocName -> Bool  
eq lemma1(S,A,D) = 
[A,apply] /in docTrace(doc(S,D)) implies docCreator(doc(S,D)) = A .
-- for proof inv3
op lemma2 : Sys ActorName ActorName DocName -> Bool  
eq lemma2(S,A1,A2,D) = 

mod ISTEP {  
pr(INV)
-- arbitrary objects  
ops s s' : -> Sys  

-- declare predicates to prove in inductive step  
op istep1 : ActorName DocName -> Bool  
opt istep2 : ActorName DocName -> Bool  
opt istep3 : ActorName ActorName DocName -> Bool  
opt istep7 : ActorName DocName -> Bool  
opt isteplemma1 : ActorName DocName -> Bool  
opt isteplemma2 : ActorName ActorName DocName -> Bool  

vars A, A1, A2 : ActorName  
vars D, D1, D2 : DocName  

-- define predicates to prove inductive step  
eq istep1(A,D) = inv1(s,A,D) implies inv1(s',A,D) .  
eq istep2(A,D) = inv2(s,A,D) implies inv2(s',A,D) .  
eq istep3(A1,A2,D) = inv3(s,A1,A2,D) implies inv3(s',A1,A2,D) .  
eq istep7(A,D) = inv7(s,A,D) implies inv7(s',A,D) .  
eq isteplemma1(A,D) = lemma1(s,A,D) implies lemma1(s',A,D) .  
eq isteplemma2(A1,A2,D) = lemma2(s,A1,A2,D) implies lemma2(s',A1,A2,D) .  
}
Appendix D

Case Studies with Transparency

Discussion

D.1 OTS/CafeOBJ Specifications of Social Welfare Case

--- CafeOBJ specification for a social welfare case
--- in Public Administration System.

--> *************************
--> Part I: General modules
--> *************************

mod! SET (P :: TRIV)
{ [Elt.P < Set]
pr(EQL)

op empty : -> Set
op add : Elt.P Set -> Set
op sub : Elt.P Set -> Set
op _/in_ : Elt.P Set -> Bool
--
op _U_ : Set Set -> Set
op _&_ : Set Set -> Set
--
op subset : Set Set -> Bool
--
vars P P1 P2 : Elt.P
vars S S1 S2 : Set
--
eq P1 /in empty = false .
eq P2 /in add(P1, S) = if (P2 = P1) then true else (P2 /in S) fi .
eq sub(P1, empty) = empty .
eq sub(P1, add(P2, S)) = (if (P1 = P2) then sub(P1, S) else add(P2, sub(P1, S)) fi) .
eq P1 /in (S1 U S2) = (P1 /in S1) or (P1 /in S2) .
eq P1 /in (S1 & S2) = (P1 /in S1) and (P1 /in S2) .
--
eq subset(empty, S) = true .
eq subset(add(P1, S1), S) = if P1 /in S then subset(S1, S) else false fi .
}

mod! LIST(X :: TRIV)
{ pr(EQL)
[Nil Elt.X < NList < List]
-- basic definition of List
op nil : -> Nil
op add : Elt.X List -> NList

123
op car_ : NList -> Elt.X
op cdr_ : NList -> List

var NnL : NList
var N : Elt.X

eq (nil = NnL) = false.
eq car(add(N, nil)) = N.
eq cdr(add(N, nil)) = nil.
eq car(add(N, NnL)) = N.
eq cdr(add(N, NnL)) = NnL.

-- extend operation on List for Document
op _/in_ : Elt.X List -> Bool
op sub : Elt.X List -> List

vars E E' : Elt.X
var L : List

-- define operation to check if E in List
eq E /in nil = false.
eq E /in add(E, nil) = true.
eq E /in add(E', L) = E /in L or (E = E').

-- define sub an element from list
eq sub(E, nil) = nil.
eq sub(E, add(E, nil)) = nil.
eq sub(E, add(E', NnL)) = (if (E = E') then sub(E, NnL) else add(E', sub(E, NnL)) fi).

mod! PAIR(M :: TRIV, N :: TRIV)

pr(EQL)
[Pair]
--
op [_,_] : Elt.M Elt.N -> Pair
op left : Pair -> Elt.M
op right : Pair -> Elt.N

var P : Pair
vars X1 X2 : Elt.M
vars Y1 Y2 : Elt.N
eq left([X1, Y1]) = X1.
eq right([X1, Y1]) = Y1.
eq ([X1, Y1] = [X2, Y2]) = (X1 = X2) and (Y1 = Y2).

mod! TRI(M :: TRIV, N :: TRIV, O :: TRIV)

pr(EQL)
[Tri]
--
op left : Tri -> Elt.M
op mid : Tri -> Elt.N
op right : Tri -> Elt.O
op _=_ : Tri Tri -> Bool

var T : Tri
vars X1 X2 : Elt.M
vars Y1 Y2 : Elt.N
vars Z1 Z2 : Elt.O
eq left([X1, Y1, Z1]) = X1.
eq mid([X1, Y1, Z1]) = Y1.
eq right([X1, Y1, Z1]) = Z1.
eq ([X1, Y1, Z1] = [X2, Y2, Z2]) = (X1 = X2) and (Y1 = Y2) and (Z1 = Z2).

--> ******************************
--> Part II : modules for Actors
--> ****************************

mod! ACTORSTATUS {

}
[ActorStatus]
pr(EQL)

ops handling idling : -> ActorStatus
--
eq (handling = idling) = false .
}

mod! ACTORROLESET {
pr (SET(P <= view to NAME sort Elt -> ActorName)
    *sort Set -> ActorRoleSet)
--
-- three roles of the public administration system
ops applicantSet agenciesSet : -> ActorRoleSet
--
op _=_ : ActorRoleSet ActorRoleSet -> Bool comm
var ARS : ActorRoleSet
eq (ARS = ARS) = true .
--
eq (applicantSet = agenciesSet) = false .
}

mod! OPERATION
[OpConst < Operation]
pr(EQL)

ops apply verify decide : -> OpConst
ops report issue track : -> OpConst
--
vars Op1 Op2 : OpConst
eq (Op1 = Op2) = (Op1 == Op2) .

mod! AUTHORIZATION
[Mand Auth < Authorization]
pr(EQL + OPERATION + ACTORROLESET)

op auth : Name Operation ActorRoleSet -> Auth
op mand : Name Operation ActorRoleSet -> Mand
--
op nameAuth : Authorization -> Name
op opAuth : Authorization -> Operation
op nameSetAuth : Authorization -> ActorRoleSet
--
vars N N1 N2 : Name
vars O O1 O2 : Operation
vars A A1 A2 : ActorRoleSet
vars AU1 AU2 : Auth
vars M1 M2 : Mand
--
eq nameAuth(auth(N,O,A)) = N .
eq opAuth(auth(N,O,A)) = O .
eq nameSetAuth(auth(N,O,A)) = A .
--
eq nameAuth(mand(N,O,A)) = N .
eq opAuth(mand(N,O,A)) = O .
eq nameSetAuth(mand(N,O,A)) = A .
--
eq (AU1 = AU2)
  = (nameAuth(AU1) = nameAuth(AU2) and opAuth(AU1) = opAuth(AU2)
   and nameSetAuth(AU1) = nameSetAuth(AU2)) .
eq (M1 = M2)
  = (nameAuth(M1) = nameAuth(M2) and opAuth(M1) = opAuth(M2)
   and nameSetAuth(M1) = nameSetAuth(M2)) .
eq (auth(N1,O1,A1) = mand(N2,O2,A2)) = false .
}

mod! OPSET {

125
\begin{verbatim}
pr (SET(P <= view to OPERATION { sort Elt -> Operation }))
    *(sort Set -> OpSet))
}

mod! AUTHSET {
    pr (SET (P <= view to AUTHORIZATION { sort Elt -> Authorization }))
        *(sort Set -> AuthSet))
}

mod! DOCSTATUS {
    pr(EQL)
    [DocStatus]
    --
    ops operating exiting : -> DocStatus
    --
    eq (operating = exiting) = false .
}

mod! DOCPAIR {
    pr (PAIR(M <= view to NAME sort Elt -> ActorName,
        N <= view to OPERATION { sort Elt -> Operation })
            *(sort Pair -> DocPair) )
}

mod! DOCTRACE {
    pr(LIST(X <= view to DOCPAIR { sort Elt -> DocPair}))
        *(sort List -> DocTrace ,
            op nil -> ndt })
    pr(OPERATION)
    op emptyDocPair : -> DocPair
    op getTerm : Operation DocTrace -> DocPair
    --
    var O : Operation
    var A : ActorName
    var DP : DocPair
    var DT : DocTrace
    --
    eq getTerm(O, ndt) = emptyDocPair .
    eq getTerm(O,add(DP,DT)) = (if O = right(DP) then DP else getTerm(O,DT) fi) .
}

mod! DOCLABEL {
    pr(EQL)
    [DocLabelConst < DocLabel]
    --
    ops applied verified decided : -> DocLabel
    ops reported issused : -> DocLabel
    op emptied : -> DocLabel
    --
    vars DLc1 DLc2 : DocLabelConst
    eq (DLc1 = DLc2) = (DLc1 == DLc2) .
    eq (applied = verified) = false .
    eq (applied = decided) = false .
    eq (applied = reported) = false .
    eq (applied = issused) = false .
    eq (applied = emptied) = false .
    eq (verified = decided) = false .
    eq (verified = reported) = false .
    eq (verified = issused) = false .
    eq (verified = emptied) = false .
    eq (decided = reported) = false .
    eq (decided = issused) = false .
    eq (decided = emptied) = false .
    eq (reported = issused) = false .
    eq (issused = emptied) = false .
}

mod! ACTOR {
    pr(EQL + NAME + ACTORSTATUS + DOCNAMESET + DOCLABEL)
    [Actor]
\end{verbatim}
op mk-actor : ActorName ActorStatus DocNameSet DocLabel -> Actor
op actorName : Actor -> ActorName
op actorStatus : Actor -> ActorStatus
op actorKnow : Actor -> DocNameSet
op actorTrack : Actor -> DocLabel
op _=_ : Actor Actor -> Bool comm

--
vars A1 A2 : Actor
var N : ActorName
var S : ActorStatus
var DS : DocNameSet
var DL : DocLabel
--
eq actorName(mk-actor(N,S,DS,DL)) = N .
eq actorStatus(mk-actor(N,S,DS,DL)) = S .
eq actorKnow(mk-actor(N,S,DS,DL)) = DS .
eq actorTrack(mk-actor(N,S,DS,DL)) = DL .
--
eq (A1 = A2) = actorName(A1) = actorName(A2) .

---> Part III : modules for Documents

mod! DOCAUTH {
  pr (LIST(X <= view to AUTHORIZATION {sort Elt -> Authorization}))
  * { sort List -> DocAuth ,
  op nil -> nda } )

  op del : Authorization DocAuth -> DocAuth
  --
  vars A1 A2 : Authorization
  var DA : DocAuth
  --
  eq del(A1,nda) = nda .
  eq del(A1,add(A2,DA)) = (if (A1 = A2) then del(A1,DA) else add(A2,del(A1,DA)) fi) .
}

mod! DOCTYPE {
  pr(EQL)
  [DocType]
  --
  ops pdoc ddoc ldoc : -> DocType
  --
  eq (pdoc = ddoc) = false .
  eq (pdoc = ldoc) = false .
  eq (ddoc = ldoc) = false .
}

---> model information in document

mod! INFODATA {
  pr(EQL + NAT)
  [InfoData]
  --
  op _=_ : InfoData InfoData -> Bool {comm}
  op _-_ : InfoData InfoData -> Nat
  -->
  *** added part
  op _<_ : InfoData InfoData -> Bool {comm}
  -->
  *** added end
  --
  vars IDATA IDATA1 IDATA2 : InfoData
  --
  eq (IDATA = IDATA1) = true .
}
mod! INFOEDUCATION {
    [InfoEdu]
    ops elementary middleschool highschool
         college university master phd : -> InfoEdu
}

mod! INFOGENDER {
    [InfoGender]
    ops female male : -> InfoGender
}

mod! INFOAGE {
    pr(NAT)
    [InfoAge]
    -- !!need semantic
    op isseniorcitizen : InfoAge -> Bool
}

mod! INFOSUBJECT {
    [InfoSubject]
    ops insocialwelfare nonsocialwelfare : -> InfoSubject
    op _=_ : InfoSubject InfoSubject -> Bool
    --
    eq (insocialwelfare = nonsocialwelfare) = false .
}

mod! INFOKEYWORDS {
    [InfoKeywords]
    --
    ops null generalkey : -> InfoKeywords
    op _=_ : InfoKeywords InfoKeywords -> Bool comm
    op _@_ : InfoKeywords InfoKeywords -> InfoKeywords
    op _<_ : InfoKeywords InfoKeywords -> Bool comm
    --
    vars IKEY IKEY1 IKEY2 : InfoKeywords
    --
    eq (null = generalkey) = false .
    eq (IKEY = IKEY) = true .
    --
    eq (null @ null) = null .
    eq (null @ IKEY) = IKEY .
    --
    eq (null < IKEY) = true .
    eq (IKEY < generalkey) = true .
    eq (IKEY1 @ IKEY2) = if IKEY1 < IKEY2 then IKEY2 else IKEY1 fi .
}

mod! INFOCONTACTFORM {
    [InfoContactForm]
    ops telephone email postoffice visit : -> InfoContactForm
}

mod! INFORESPOND {
    [InfoRespond]
    ops nonrespond respond : -> InfoRespond
}

mod! INFORESULT {
    pr(EQL)
    [InfoResult]
}
ops pass unpass undeceive : -> InfoResult
--
eq (pass = unpass) = false .
eq (pass = undeceive) = false .
eq (unpass = undeceive) = false .
}

mod! INFORMATION {
[Info]
pr(EQL + INFOEDATAC + INFOEDUCATION + INFOGENDER + INFOAGE )
pr(INFOSUBJECT + INFOKEYWORDS + INFOCONTACTFORM + INFORESPOND + INFORESULT)

ops none general noenough : -> Info
op mk-info : InfoData InfoEdu InfoGender InfoAge InfoSubject
            InfoKeywords InfoContactForm InfoRespond InfoResult -> Info
op infoData : Info -> InfoData
op infoEdu : Info -> InfoEdu
up infoGender : Info -> InfoGender
up infoAge : Info -> InfoAge
up infoSubject : Info -> InfoSubject
up infoKeywords : Info -> InfoKeywords
up infoContactForm : Info -> InfoContactForm
up infoRespond : Info -> InfoRespond
up infoResult : Info -> InfoResult
--
op _=_ : Info Info -> Bool comm
op @_ : Info Info -> Info
-- *** added part
op _<_ : Info Info -> Bool comm
-- *** added end
--
vars I I1 I2 : Info
--
eq (none = general) = false .
eq (I = I) = true .
--
eq (none @ none) = none .
eq (none @ I) = I .
--
eq (none < I) = true .
eq (I < general) = true .
eq (I1 @ I2) = if I1 < I2 then I2 else I1 fi .
}

mod! DOCUMENT {
[Application < Doc]
pr(EQL + NAME + DOCSTATUS + DOCTYPE + INFORMATION + DOCNAMESET + DOCTRACE + DOCAUTH + DOCLABEL)

op mk-doc : DocName ActorName DocStatus DocType Info
            DocTrace DocAuth DocNameSet DocLabel -> Doc
op _-_ : Doc Doc -> Bool comm
--
-- constant "emptyDoc" is used to denote the initial value of a doc that is not created yet.
op emptyDoc : -> Doc
-- constant "criteria" is used to denote the initial value of a doc in basicLawSet
op criteria : -> Doc
--

op docName : Doc -> DocName
op docCreator : Doc -> ActorName
op docStatus : Doc -> DocStatus
op docType : Doc -> DocType
op docInfo : Doc -> Info
up docTrace : Doc -> DocTrace
up docAuth : Doc -> DocAuth
up docRefer : Doc -> DocNameSet
up docLabel : Doc -> DocLabel
--
vars D D1 D2 : Doc
vars N N1 N2 : DocName
var A : ActorName
var S : DocStatus
var T : DocType
var I : Info
var DT : DocTrace
var DA : DocAuth
var DS : DocNameSet
var DL : DocLabel
var ID : InfoData
var IA : InfoAge
var IE : InfoEdu
var IG : InfoGender
var IK : InfoKeywords
var ICF : InfoContactForm
var IR : InfoRespond
var IRT : InfoResult
-- op SW : -> DocName .
--
eq (docName(criteria) /in basicLawSet) = true .
eq (docCreator(criteria) = A) = false .
eq docTrace(criteria) = ndt .
eq docLabel(criteria) = issused .
-- use for defining the logic expression
eq docInfo(criteria) = mk-info(ID, IE, IG, IA, insocialwelfare, IK, if isseniorcitizen(IA) then visit else ICF fi,
IR, IRT) .
eq docLabel(emptyDoc) = emptied .
eq (docName(emptyDoc) = N) = false .
eq (docCreator(emptyDoc) = A) = false .
eq docTrace(emptyDoc) = ndt .
-- > ** added part -- eq (docInfo(D) < docInfo(basicLaw)) = true .
eq (docInfo(emptyDoc) < docInfo(D)) = true .
-- > ** added end
--
eq docName(mk-doc(N,A,S,T,I,DT,DA,DS,DL)) = N .
eq docCreator(mk-doc(N,A,S,T,I,DT,DA,DS,DL)) = A .
eq docType(mk-doc(N,A,S,T,I,DT,DA,DS,DL)) = T .
eq docInfo(mk-doc(N,A,S,T,I,DT,DA,DS,DL)) = I .
eq docTrace(mk-doc(N,A,S,T,I,DT,DA,DS,DL)) = DT .
eq docRefer(mk-doc(N,A,S,T,I,DT,DA,DS,DL)) = DS .
--
eq (D1 = D2) = (docName(D1) = docName(D2)) .
-- the following function checks if there exists at least one document whose label is issused
op hasiss : DocNameSet -> Bool
  eq hasiss(emptyDocNameSet) = false .
eq hasiss(docName(D)) = (if docLabel(D) = issused then true else false fi) .
eq hasiss(add(docName(D), DS)) = (if docLabel(D) = issused then true else hasiss(DS) fi) .
}

-- part IV : Public Administration System
--> ***************************************
mod* PA-SW {
  pr (ACTOR + DOCUMENT + AUTHSET + OPSET + ACTORROLESET + INFORESULT)
  pr (EQL)
  *[Sys1]*

  -- initial state
  op init : -> Sys1

  -- observer
  bop actor : Sys1 ActorName -> Actor
  bop doc : Sys1 DocName -> Doc

  -- actions
  bop apply : Sys1 ActorName DocName Info -> Sys1

130
bop verify : Sys1 ActorName DocName DocNameSet Info -> Sys1
bop decide : Sys1 ActorName DocName Info -> Sys1
bop issue : Sys1 ActorName DocName Info -> Sys1
bop track : Sys1 ActorName DocName -> Sys1

-- variable
var S : Sys1
vars C C1 C2 : ActorName
vars D D1 D2 : DocName
vars DS DS1 DS2 : DocNameSet
var I : Info

--> added operation
op sumInfo : DocNameSet -> Info
--
eq sumInfo(add(D,emptyDocNameSet)) = docInfo(doc(S,D)) .
eq sumInfo(add(D,DS)) = (docInfo(doc(S,D)) @ sumInfo(DS)) .

--> added end

-- [0] equations of initial state
eq actor(init,C) = mk-actor(C,idling,basicLawSet,issused) .
eq doc(init,D) = (if D /in basicLawSet then criteria else emptyDoc fi) .

-- [1] apply
op c-apply : Sys1 ActorName DocName Info -> Bool
eq c-apply(S,C,D,I) = doc(S,D) = emptyDoc and C /in applicantSet and not (D /in basicLawSet) .
--
ceq actor(apply(S,C,D,I),C1) = (if not(C1 = C) then actor(S,C1)
else mk-actor(C,handling,actorKnow(actor(S,C)),actorTrack(actor(S,C))) fi)
if c-apply(S,C,D,I) .
ceq doc(apply(S,C,D,I),D1) = (if not(D1 = D) then doc(S,D1)
else mk-doc(D,C,operating,pdoc,I,
add([C,apply],ndt),auth(docCreator(doc(S,D)),verify,agenciesSet),
actorKnow(actor(S,C),applied) fi) if c-apply(S,C,D,I) .
ceq apply(S,C,D,I) = S if not c-apply(S,C,D,I) .

-- [2] verify
op c-verify : Sys1 ActorName DocName DocNameSet Info -> Bool
eq c-verify(S,C,D,DS,I) = C /in agenciesSet and not (C = docCreator(doc(S,D))) and
docType(doc(S,D)) = pdoc and docLabel(doc(S,D)) = applied and
(auth(docCreator(doc(S,D)),verify,agenciesSet) /in docAuth(doc(S,D))) .
--
ceq actor(verify(S,C,D,DS,I),C1) = (if not(C1 = C) then actor(S,C1)
else mk-actor(C,handling,actorKnow(actor(S,C)),actorTrack(actor(S,C))) fi)
if c-verify(S,C,D,DS,I) .
ceq doc(verify(S,C,D,DS,I),D1) = (if not(D1 = D) then doc(S,D1)
else mk-doc(D,C,operating,pdoc,I,
add([C,verify],ndt),auth(docCreator(doc(S,D)),verify,agenciesSet),
actorKnow(actor(S,C),applied) fi) if c-apply(S,C,D,I) .
ceq verify(S,C,D,DS,I) = S if not c-verify(S,C,D,DS,I) .

-- [3] decide
op c-decide : Sys1 ActorName DocName Info -> Bool
eq c-decide(S,C,D,I) = C /in agenciesSet and not(C = docCreator(doc(S,D)))
and not(C = left(getTerm(verify,docTrace(doc(S,D))))) and
-- docType(doc(S,D)) = pdoc and docLabel(doc(S,D)) = verified and
   (auth(left(getTerm(verify,docTrace(doc(S,D)))),decide,agenciesSet) /in docAuth(doc(S,D))) .

--
ceq actor(decide(S,C,D,I),C1)
   = (if not(C1 = C) then actor(S,C1)
      else mk-actor(C1,handling,actorKnow(actor(S,C1)),actorTrack(actor(S,C1))) fi) if c-decide(S,C,D,I) .

ceq doc(decide(S,C,D,I),D1)
   = (if not(D1 = D) then doc(S,D1)
      else mk-doc(D,docCreator(doc(S,D)),docStatus(doc(S,D)),docType(doc(S,D)),
                   (I @ docInfo(doc(S,D))),add([C,decide],docTrace(doc(S,D))),
                   del(auth(left(getTerm(verify,docTrace(doc(S,D)))),decide,agenciesSet),
                    (if I = noenough then docAuth(doc(S,D))
                     else add(auth(C,report,agenciesSet),docAuth(doc(S,D))) fi)),
                   (actorKnow(actor(S,C)) U docRefer(doc(S,D))),applied) fi) if c-decide(S,C,D,I) .

ceq decide(S,C,D,I) = S if not c-decide(S,C,D,I) .

op c-report : Sys1 ActorName DocName Info -> Bool
eq c-report(S,C,D,I)
   = C /in agenciesSet and not(C = docCreator(doc(S,D))) and
     not(C = left(getTerm(decide,docTrace(doc(S,D))))) and
   docLabel(doc(S,D)) = decided and
   auth(left(getTerm(decide,docTrace(doc(S,D)))),report,agenciesSet) /in docAuth(doc(S,D)) .

--
ceq actor(report(S,C,D,I),C)
   = (if not(C1 = C) then
      (if C1 = docCreator(doc(S,D))
       then mk-actor(C1,idling,add(D,actorKnow(actor(S,C1))),actorTrack(actor(S,C1)))
       else actor(S,C1) fi)
      else mk-actor(C,handling,actorKnow(actor(S,C)),actorTrack(actor(S,C))) fi)
   if c-report(S,C,D,I) .

ceq doc(report(S,C,D,I),D)
   = (if not(D1 = D) then doc(S,D1)
      else mk-doc(D,docCreator(doc(S,D)),docStatus(doc(S,D)),docType(doc(S,D)),
                   docInfo(doc(S,D)),add([C,report],docTrace(doc(S,D))),
                   del(auth(left(getTerm(decide,docTrace(doc(S,D)))),report,agenciesSet),
                    (if (infoResult(I) = unpass) then docAuth(doc(S,D))
                     else add(auth(C,issuse,agenciesSet),docAuth(doc(S,D))) fi)),
                   docRefer(doc(S,D)),reported)) fi) if c-report(S,C,D,I) .

ceq report(S,C,D,I) = S if not c-report(S,C,D,I) .

-- [5] issuse
op c-issuse : Sys1 ActorName DocName Info -> Bool
eq c-issuse(S,C,D,I)
   = C /in agenciesSet and not(C = docCreator(doc(S,D))) and
     not(C = left(getTerm(decide,docTrace(doc(S,D))))) and
   docType(doc(S,D)) = ddoc and docLabel(doc(S,D)) = reported and
   auth(left(getTerm(decide,docTrace(doc(S,D)))),issuse,agenciesSet) /in docAuth(doc(S,D)) .

--
ceq actor(issuse(S,C,D,I),C)
   = (if not(C1 = C) then actor(S,C)
      else mk-actor(C,handling,actorKnow(actor(S,C)),actorTrack(actor(S,C))) fi)
   if c-issuse(S,C,D,I) .

ceq doc(issuse(S,C,D,I),D)
   = (if not(D1 = D) then doc(S,D)
      else mk-doc(D,docCreator(doc(S,D)),exiting,docType(doc(S,D)),
                   docInfo(doc(S,D)),add([C,issuse],docTrace(doc(S,D))),
                   del(auth(left(getTerm(report,docTrace(doc(S,D))))),issuse,agenciesSet),
                   docAuth(doc(S,D)),docRefer(doc(S,D)),issused) fi)
   if c-issuse(S,C,D,I) .

ceq issuse(S,C,D,I) = S if not c-issuse(S,C,D,I) .

-- [6] track
op c-track : Sys1 ActorName DocName -> Bool
eq c-track(S,C,D)

132
- \((C = \text{docCreator}(\text{doc}(S,D)))\) and not \((\text{doc}(S,D) = \text{emptyDoc})\) or \((D \in \text{basicLawSet})\).

```latex
\text{ceq actor(track(S,C,D),C1) = (if not(C1 = C) then actor(S,C1) else mk-actor(C1,handling,actorKnow(actor(S,C1)), docLabel(doc(S,D))) fi) if c-track(S,C,D).}
```

```latex
\text{ceq doc(track(S,C,D),D1) = (if not (D = D1) then doc(S,D1) else mk-doc(D1, C, docStatus(doc(S,D1)),docType(doc(S,D1)),
\text{docInfo(doc(S,D1)),add([C,track], \text{docTrace}(doc(S,D1))),docAuth(doc(S,D1)),
\text{docRefer(doc(S,D1)), docLabel(doc(S,D1))) fi) if c-track(S,C,D).}
```

```latex
\text{ceq track(S,C,D) = S if not c-track(S,C,D).}
```

### D.2 Refinement Proof

\text{mod INV { }
p_{\text{pr}(\text{PA-GN} + \text{PA-SW})}

\text{ops a a0 a1 a2 a3 : -> ActorName}
\text{ops d d0 d1 d2 d3 : -> DocName}
\text{op i : -> Info}
\text{op r : -> ResultGn}

---

\text{-- anti-corruption}
\text{-- Generic model}
\text{op GN-inv1 : Sys0 ActorName DocName -> Bool}
\text{op GN-inv2 : Sys0 ActorName DocName -> Bool}
\text{op GN-lemma1 : Sys0 ActorName DocName -> Bool}
\text{op GN-lemma2 : Sys0 ActorName ActorName DocName -> Bool}

\text{-- Social Welfare}
\text{op SW-inv1 : Sys1 ActorName DocName -> Bool}
\text{op SW-inv2 : Sys1 ActorName DocName -> Bool}
\text{op SW-inv3 : Sys1 ActorName ActorName DocName -> Bool}

---

\text{-- CafeOBJ variables}
\text{var S0 : Sys0}
\text{var S1 : Sys1}
\text{vars A1 A2 A : ActorName}
\text{vars D1 D2 D : DocName}

---

\text{-- From the idea of anti-corruption}

---

\text{-- Defining invariants for GN}
\text{eq GN-inv1(S0,A,D) = [A,evaluate] /in docTrace(doc(S0,D)) or [A,confirm] /in docTrace(doc(S0,D))
\text{implies not([A,apply] /in docTrace(doc(S0,D)))}.}
\text{eq GN-inv2(S0,A,D) = [A,execute] /in docTrace(doc(S0,D)) implies not([A,apply] /in docTrace(doc(S0,D))}.}
\text{eq GN-inv3(S0,A1,A2,D) = [A1,confirm] /in docTrace(doc(S0,D)) and [A2,execute] /in docTrace(doc(S0,D)) implies not(A1 = A2).}
\text{eq GN-lemma1(S0,A,D) = [A,apply] /in docTrace(doc(S0,D)) implies docCreator(doc(S0,D)) = A}.}

---

\text{-- for proof inv3}
\text{eq GN-lemma2(S0,A1,A2,D) = [A1,execute] /in docTrace(doc(S0,D)) implies [A2,confirm] /in docTrace(doc(S0,D))}.}

---

\text{-- lemmas of GN [the lemmas are used in "bl2gn-sim1-proof.mod"]}
\text{--> proved (no other lemmas used)}
\text{op GN-lemma3 : Sys0 DocName -> Bool}
\text{op GN-lemma3(S0,D) = docLabel(doc(S0,D)) = applied implies
\text{(docType(doc(S0,D))) = pdoc and
\text{auth(docCreator(doc(S0,D)),evaluate,agenciesSet) /in docAuth(doc(S0,D))})}.}

---

\text{-- proved (no other lemmas used)}
\text{op GN-lemma4 : Sys0 DocName -> Bool}
```
eq GN-lemma4(S₀,D) \\
  = docLabel(doc(S₀,D)) = evaluated implies \\
  (docType(doc(S₀,D)) = pdoc and \\
   auth(left(getTerm(evaluate,docTrace(doc(S₀,D))))),confirm,agenciesSet) /in docAuth(doc(S₀,D))) .

--> proved (no other lemmas used)

op GN-lemma5 : Sys₀ DocName → Bool 

eq GN-lemma5(S₀,D) \\
  = docLabel(doc(S₀,D)) = confirmed implies \\
  (auth(left(getTerm(confirm,docTrace(doc(S₀,D))))),notify,agenciesSet) /in docAuth(doc(S₀,D))) .

--> proved (no other lemmas used)

op GN-lemma6 : Sys₀ DocName → Bool 

eq GN-lemma6(S₀,D) \\
  = docLabel(doc(S₀,D)) = notified implies \\
  (docType(doc(S₀,D)) = ddoc and \\
   auth(left(getTerm(notify,docTrace(doc(S₀,D))))),execute,agenciesSet) /in docAuth(doc(S₀,D))) .

--> proved (no other lemmas used)

op GN-lemma7 : Sys₀ DocName → Bool 

eq GN-lemma7(S₀,D) \\
  = (docLabel(doc(S₀,D)) = applied or \\
   docLabel(doc(S₀,D)) = evaluated or \\
   docLabel(doc(S₀,D)) = confirmed or \\
   docLabel(doc(S₀,D)) = notified or \\
   docLabel(doc(S₀,D)) = executed) implies not(doc(S₀,D) = emptyDocGn) .

-- Defining invariants for SW

eq SW-inv1(S₁,A,D) \\
  = [A,verify] /in docTrace(doc(S₁,D)) or [A,decide] /in docTrace(doc(S₁,D)) \\
   implies not([A,apply] /in docTrace(doc(S₁,D))) .

eq SW-inv2(S₁,A,D) \\
  = [A,issue] /in docTrace(doc(S₁,D)) implies not([A,apply] /in docTrace(doc(S₁,D))) .

eq SW-inv3(S₁,A₁,A₂,D) \\
  = [A₁,decide] /in docTrace(doc(S₁,D)) and [A₂,issue] /in docTrace(doc(S₁,D)) implies not(A₁ = A₂) .

-- lemmas of BL [the lemmas are used in "bl2gn-inv1-proof.mod"]

--> proved (no other lemmas used)

op SW-lemma1 : Sys₁ ActorName DocName → Bool 

eq SW-lemma1(S₁,A,D) \\
  = docLabel(doc(S₁,D)) = appliedBl implies not([A,verify] /in docTrace(doc(S₁,D)))) .

--> proved (no other lemmas used)

op SW-lemma2 : Sys₁ ActorName DocName → Bool 

eq SW-lemma2(S₁,A,D) \\
  = docLabel(doc(S₁,D)) = appliedBl implies not([A,decide] /in docTrace(doc(S₁,D)))) .

--> proved (with SW-lemma2)

op SW-lemma3 : Sys₁ ActorName DocName → Bool 

eq SW-lemma3(S₁,A,D) \\
  = docLabel(doc(S₁,D)) = appliedBl implies not([A,decide] /in docTrace(doc(S₁,D)))) .

--> this can be proved, but this makes the refinement proof not meaningful.

--> proved (with SW-lemma6)

op SW-lemma4 : Sys₁ ActorName ActorName DocName → Bool 

eq SW-lemma4(S₁,A₁,A₂,D) \\
  = ([A₁,verify] /in docTrace(doc(S₁,D)) and [A₂,apply] /in docTrace(doc(S₁,D))) implies not(A₁ = A₂) .

--> this can be proved, but this makes the refinement proof not meaningful.

--> proved (with SW-lemma6)

op SW-lemma5 : Sys₁ ActorName ActorName DocName → Bool 

eq SW-lemma5(S₁,A₁,A₂,D) \\
  = ([A₁,decide] /in docTrace(doc(S₁,D)) and [A₂,apply] /in docTrace(doc(S₁,D))) implies not(A₁ = A₂) .

--> proved (no other lemmas used)

op SW-lemma6 : Sys₁ ActorName DocName → Bool 

eq SW-lemma6(S₁,A,D) \\
  = [A,apply] /in docTrace(doc(S₁,D)) implies docCreator(doc(S₁,D)) = A .
mod ISTEP { 
pr(INV) 
-- arbitrary objects 
ops s0 s0' : -> Sys0 
ops s1 s1' : -> Sys1 

-- declare predicates to prove in inductive step 
op GN-istep1 : ActorName DocName -> Bool 
op GN-istep2 : ActorName DocName -> Bool 
op GN-istep3 : ActorName ActorName DocName -> Bool 
op GN-isteplemma1 : ActorName DocName -> Bool 
op GN-isteplemma2 : ActorName ActorName DocName -> Bool 
op GN-isteplemma3 : DocName -> Bool 
op GN-isteplemma4 : DocName -> Bool 
op GN-isteplemma5 : DocName -> Bool 
op GN-isteplemma6 : DocName -> Bool 
op GN-isteplemma7 : DocName -> Bool 
op SW-istep1 : ActorName DocName -> Bool 
op SW-istep2 : ActorName DocName -> Bool 
op SW-istep3 : ActorName ActorName DocName -> Bool 
op SW-istepl1 : ActorName DocName -> Bool 
op SW-istepl2 : ActorName DocName -> Bool 
op SW-istepl3 : ActorName DocName -> Bool 
op SW-istepl4 : ActorName ActorName DocName -> Bool 
op SW-istepl5 : ActorName ActorName DocName -> Bool 
op SW-istepl6 : ActorName DocName -> Bool 

vars A A1 A2 : ActorName 
vars D D1 D2 : DocName 

-- define predicates to prove inductive step 
eq GN-istep1(A,D) = GN-inv1(s0,A,D) implies GN-inv1(s0',A,D) . 
eq GN-istep2(A,D) = GN-inv2(s0,A,D) implies GN-inv2(s0',A,D) . 
eq GN-istep3(A1,A2,D) = GN-inv3(s0,A1,A2,D) implies GN-inv3(s0',A1,A2,D) . 
eq GN-isteplemma1(A,D) = GN-lemma1(s0,A,D) implies GN-lemma1(s0',A,D) . 
eq GN-isteplemma2(A1,A2,D) = GN-lemma2(s0,A1,A2,D) implies GN-lemma2(s0',A1,A2,D) . 
eq GN-isteplemma3(D) = GN-lemma3(s0,D) implies GN-lemma3(s0',D) . 
eq GN-isteplemma4(D) = GN-lemma4(s0,D) implies GN-lemma4(s0',D) . 
eq GN-isteplemma5(D) = GN-lemma5(s0,D) implies GN-lemma5(s0',D) . 
eq GN-isteplemma6(D) = GN-lemma6(s0,D) implies GN-lemma6(s0',D) . 
eq GN-isteplemma7(D) = GN-lemma7(s0,D) implies GN-lemma7(s0',D) . 
eq SW-istep1(A,D) = SW-inv1(s1,A,D) implies SW-inv1(s1',A, D) . 
eq SW-istep2(A,D) = SW-inv2(s1,A,D) implies SW-inv2(s1',A,D) . 
eq SW-istep3(A1,A2,D) = SW-inv3(s1,A1,A2,D) implies SW-inv3(s1',A1,A2,D) . 
eq SW-istepl1(A,D) = SW-lemma1(s1,A,D) implies SW-lemma1(s1',A,D) . 
eq SW-istepl2(A,D) = SW-lemma2(s1,A,D) implies SW-lemma2(s1',A,D) . 
eq SW-istepl3(A,D) = SW-lemma3(s1,A,D) implies SW-lemma3(s1',A,D) . 
eq SW-istepl4(A1,A2,D) = SW-lemma4(s1,A1,A2,D) implies SW-lemma4(s1',A1,A2,D) . 
eq SW-istepl5(A1,A2,D) = SW-lemma5(s1,A1,A2,D) implies SW-lemma5(s1',A1,A2,D) . 
eq SW-istepl6(A,D) = SW-lemma6(s1,A,D) implies SW-lemma6(s1',A,D) . 
}
Appendix E

Specification of PGLL

```
-- CafeOBJ model for PGLL

--> **********************
--> General Model
--> **********************

-- standard basic List
mod! LIST (X :: TRIV) {
  pr ( EQL )
  [ Nil NnList < List ] -- Nil Non-nil-List < List
  op nil : -> Nil
  op cons : Elt List -> NnList
  op hd : NnList -> Elt -- head of the list
  op tl : List -> List -- tail of the list

  vars E E' : Elt
  var L : List
  var NnL : NnList

  eq hd(cons(E, L)) = E .
  eq tl(nil) = nil .
  eq tl(cons(E, L)) = L .

  -- define inequality
  eq (nil = NnL) = false .
}

-- enhanced List for license
mod! LIST+ {
  pr ( LIST * { op cons -> _;_ } )
    -- rename operator cons to _;_ for convenience
  pr ( NAT ) -- NAT is a built-in system module in CafeOBJ

  op _in_ : Elt List -> Bool -- check whether an element is in a list
  op r-hd_ : NnList -> Elt -- reverse-head
  op r-tl_ : List -> List -- reverse-tail
  op length_ : List -> Nat -- count the number of elements of a list

  vars E E' : Elt
  vars L L' : List
  var NnL : NnList

  eq E in (E' ; L) = (E = E') or (E in L) .
  eq E in nil = false .

  eq r-hd(E ; NnL) = r-hd(NnL) .
  eq r-hd(E ; nil) = E .
```

eq r-tl(E ; L) = if L = nil then nil else (E ; r-tl(L)) fi .
eq r-tl(nil) = nil .
eq length(E ; L) = 1 + length(L) .
eq length(nil) = 0 .

mod! SET (P :: TRIV)

[Elit.P < Set]
pr(EQL)

op empty : -> Set
op add : Elit.P Set -> Set
op sub : Elit.P Set -> Set
op _/in_ : Elit.P Set -> Bool
--
op _U_ : Set Set -> Set
op _&_ : Set Set -> Set
-- S2 = S1
op del : Set Set -> Set
--
op subset : Set Set -> Bool
--

vars P P1 P2 : Elit.P
vars S S1 S2 : Set
--
eq P1 /in empty = false .
eq P2 /in add(P1, S) = if (P2 = P1) then true else (P2 /in S) fi .
eq sub(P1, empty) = empty .
eq sub(P1, add(P2, S)) = (if (P1 = P2) then sub(P1, S) else add(P2, sub(P1, S)) fi) .
eq P1 /in (S1 U S2) = (P1 /in S1) or (P1 /in S2) .
eq P1 /in (S1 & S2) = (P1 /in S1) and (P1 /in S2) .

eq del(empty, S2) = S2 .
eq del(add(P1, S1), S2) = del(S1, sub(P1, S2)) .
eq P1 /in (del(S1, S2)) = (P1 /in S2) and (not (P1 /in S1)) .
eq subset(empty, S) = true .
eq subset(add(P1, S1), S) = if P1 /in S then subset(S1, S) else false fi .

mod* NAME

[LicName DocName ActorName < Name]
pr(EQL)

op omit : -> LicName

mod* LICNAMESET

pr (SET(P <= view to NAME sort Elt -> LicName )
  *sort Set -> LicNameSet,
  op empty -> emptylic)
op naturelic : -> LicNameSet
}

mod* ACTOR

pr(NAME * LICNAMESET)
[Actor]

-- the components of an Actor is deliberately undefined.
op mk-actor : ActorName LicNameSet LicNameSet -> Actor
op.actorName : Actor -> ActorName
op.actorLicit : Actor -> LicNameSet
op.actorLicee : Actor -> LicNameSet

137
op _=_ : Actor Actor -> Bool comm
op nature : -> Actor

--

vars A1 A2 : Actor
vars AN1 AN2 : ActorName
vars LS1 LS2 LS3 LS4 : LicNameSet

--
eq actorName(mk-actor(AN1,LS1,LS2)) = AN1 .
eq actorLicor(mk-actor(AN1,LS1,LS2)) = LS1 .
eq actorLicee(mk-actor(AN1,LS1,LS2)) = LS2 .
eq (mk-actor(AN1,LS1,LS2) = mk-actor(AN2,LS3,LS4)) = (AN1 = AN2) and (LS1 = LS3) and (LS2 = LS4) .

mod! INFORMATION {
  [Info]
  pr(EQL)
  ops none general original : -> Info
  --
  op _=_ : Info Info -> Bool comm
  op _@_ : Info Info -> Info
  --
  vars I I1 I2 : Info
  --
  eq (none = general) = false .
eq (none = original) = false .
eq (general = original) = false .
eq (I = I) = true .
  --
eq (none @ none) = none .
eq (none @ I ) = I .
}

mod! DOCUMENT {
  [Doc]
  pr(EQL + NAME + INFORMATION + ACTOR)
  op mk-doc : DocName Actor Info -> Doc
  op _=_ : Doc Doc -> Bool comm
  op arbitrary : -> Doc
  --
  op docName : Doc -> DocName
  op creator : Doc -> Actor
  op docInfo : Doc -> Info
  --
  vars D D1 D2 : Doc
  vars DN1 DN2 DN : DocName
  var A : Actor
  var I : Info
  --
  eq docName(mk-doc(DN, A, I)) = DN .
eq creator(mk-doc(DN, A, I)) = A .
eq docInfo(mk-doc(DN, A, I)) = I .
}

mod* ACTORNAMESET {
  pr (SET(P <= view to NAME { sort Elt -> ActorName }))
  *(sort Set -> ActorNameSet)
  op allActor : -> ActorNameSet
}

mod* DOCNAMESET {
  pr (SET(P <= view to NAME { sort Elt -> Name }))
  *(sort Set -> DocNameSet)
}

mod* FUNCTION {
  [Functions]
}
mod* OPERATION {
pr ( EQL )
pr ( DOCUMENT + NAME + ACTOR + ACTORNAMESET + FUNCTION )

[Create Read Edit Copy Share Revoke Calculate Shred < Operation]

op create : Doc Doc -> Create
op read : Doc -> Read
op edit : Doc -> Edit
op copy : Doc -> Copy
op share : Doc ActorNameSet -> Share
op revoke : Doc ActorNameSet -> Revoke
op calculate : Doc Function -> Calculate
-- consider later
op shred : Doc -> Shred
op _=_ : Operation Operation -> Bool
op iscreate : Operation -> Bool
--
var C : Create
vars D1 D2 D3 D4 D : Doc
vars O1 O2 O : Operation
vars AS AS1 AS2 : ActorNameSet
vars F1 F2 F : Function
var A : Actor
var DN : DocName
--
eq doc(create(D1, D2)) = D1 .
eq doc(read(D)) = D .
eq doc(edit(D)) = D .
eq doc(copy(D1)) = D1 .
eq doc(share(D, AS)) = D .
eq doc(revoke(D, AS)) = D .
eq doc(calculate(D, F)) = D .
eq doc(shred(D)) = D .
--
eq (create(D1, D2) = create(D3, D4)) = ((D1 = D3) and (D2 = D4)) .
eq (read(D1) = read(D2)) = (D1 = D2) .
eq (edit(D1) = edit(D2)) = (D1 = D2) .
eq (copy(D1) = copy(D2)) = (D1 = D2) .
eq (share(D1, AS1) = share(D2, AS2)) = ((D1 = D2) and (AS1 = AS2)) .
eq (revoke(D1, AS1) = revoke(D2, AS2)) = ((D1 = D2) and (AS1 = AS2)) .
eq (calculate(D1, F1) = calculate(D2, F2)) = (D1 = D2 and (F1 = F2)) .
eq (shred(D1) = shred(D2)) = (D1 = D2) .
--
--
--
--
--
--

--
-> ???? no sure if the definition is correct???
eq iscreate(C) = true .
eq iscreate(create(D1,D2)) = true .
eq iscreate(create(mk-doc(DN,A,none),arbitrary)) = true .
}

mod* OPERATIONSET {
pr ( SET(P <= view to OPERATION { sort Elt -> Operation } )

*{sort Set -> OperationSet})

op all : -> OperationSet
op Operation : -> OperationSet
op mandateD : OperationSet -> OperationSet
op mandateL : OperationSet -> OperationSet
--
op iscreator : ActorName OperationSet -> Bool
op isonlycreate : OperationSet -> Bool
--
vars OS1 OS2 OS : OperationSet
var O : Operation
var A : ActorName
--
eq (mandateD(OS) U mandateL(OS)) = all .
eq mandateD(OS) & all = mandateL(OS) .
}
eq mandateL(OS) & all = mandateD(OS) .
--
eq iscreator(A, empty) = true .
eq iscreator(A, add(O, OS)) = if A = creator(doc(O)) then iscreator(A, OS) else false fi .
eq isonlycreate(add(O, OS)) = if iscreate(O) then isonlycreate(OS) else false fi .
}

mod* LICENSE {
pr(EQL)
pr(NAME + ACTOR + ACTORNAMESET + DOCUMENT + OPERATION + OPERATIONSET + DOCNAMESET)
[License]
op mk-lic : LicName ActorName ActorNameSet OperationSet -> License
--
op licname : License -> LicName
op licensor : License -> ActorName
op licensee : License -> ActorNameSet
op operationset : License -> OperationSet
--
op no-license : -> License
--
var L : License
var LN : LicName
var A : ActorName
var AS : ActorNameSet
var OS : OperationSet
--
eq liename(mk-lic(LN, A, AS, OS)) = LN .
eq licensor(mk-lic(LN, A, AS, OS)) = A .
eq licensee(mk-lic(LN, A, AS, OS)) = AS .
eq operationset(mk-lic(LN, A, AS, OS)) = OS .
eq A /in licensee(no-license) = false .
}

--> ***********************
--> PGLL
--> ***********************

mod* LICENSES {
pr(EQL)
pr(LICENSE)
*[LicSys]*

-- any initial state
op init : -> LicSys

-- observation operators
bop lic : LicSys LicName -> License
bop actor : LicSys ActorName -> Actor
bop docset : LicSys -> DocNameSet

-- transaction actions
bop grant : LicSys LicName ActorName OperationSet ActorNameSet -> LicSys
bop extend : LicSys LicName ActorName OperationSet -> LicSys
bop restrict : LicSys LicName ActorName OperationSet -> LicSys
bop withdraw : LicSys LicName ActorName -> LicSys
bop run : LicSys LicName ActorName Operation -> LicSys

--
var LS : LicSys
var L : License
var LNS : LicNameSet
vars A1 A2 A : ActorName
var AS : ActorNameSet
var DN : DocName
var N : Name
var D : Doc
var I : Info
vars LN1 LN2 LN : LicName
var O : Operation
vars OS1 OS2 OS : OperationSet
-- *** added operation
op actorOperSet : LicNameSet -> OperationSet
--
eq actorOperSet(add(LN,emptylic)) = operationset(lic(LS,LN)).
eq actorOperSet(add(LN,LNS)) = operationset(lic(LS,LN)) U actorOperSet(LNS).
-- *** added end

-- [0] init
eq lic(init,LN) = no-license.
eq actor(init,A) = mk-actor(A,emptylic,emptylic).
eq docset(init) = empty.

-- [1] grant
op c-grant : LicSys LicName ActorName OperationSet ActorNameSet -> Bool
eq c-grant(LS, LN, A1, OS, AS)
  = (lic(LS, LN) = no-license) and (iscreator(A1,OS)
or (subset(mandateL(OS),actorOperSet(actorLicee(actor(LS,A1))))).
--
ceq lic(grant(LS, LN, A1, OS, AS), LN1)
  = (if (LN1 = LN) then mk-lic(LN, A1, AS, OS)
  else lic(LS, LN1) fi) if c-grant(LS, LN, A1, OS, AS).
ceq actor(grant(LS, LN, A1, OS, AS),A)
  = (if (A = A1) then mk-actor(A,add(LN,actorLicor(actor(LS,A)))),actorLicee(actor(LS,A)))
  else if (A /in AS) then mk-actor(A,actorLicor(actor(LS,A)),add(LN,actorLicee(actor(LS,A))))
  else actor(LS,A) fi) fi) if c-grant(LS, LN, A1, OS, AS).
ceq docset(grant(LS, LN, A1, OS, AS))
  = docset(LS) if c-grant(LS, LN, A1, OS, AS).
ceq grant(LS, LN, A1, OS, AS)
  = LS if not c-grant(LS, LN, A1, OS, AS).

-- [2] extend
op c-extend : LicSys LicName ActorName OperationSet -> Bool
eq c-extend(LS, LN, A, OS)
  = ((A = licensor(lic(LS, LN))) and subset(mandateL(OS),actorOperSet(actorLicee(actor(LS,A))))
or iscreator(A,OS)).
--
ceq lic(extend(LS, LN, A, OS), LN1)
  = (if not (LN1 = LN)
  then lic(LS, LN)
  else mk-lic(LN, A, licensee(lic(LS, LN)), (OS U operationset(lic(LS, LN)))) fi) if c-extend(LS, LN, A, OS).
ceq actor(extend(LS, LN, A, OS), A1)
  = actor(LS, A1) if c-extend(LS, LN, A, OS).
ceq docset(extend(LS, LN, A, OS))
  = docset(LS) if c-extend(LS, LN, A, OS).
ceq extend(LS, LN, A, OS)
  = LS if not c-extend(LS, LN, A, OS).

-- [3] restrict
op c-restrict : LicSys LicName ActorName OperationSet -> Bool
eq c-restrict(LS, LN, A, OS)
  = (A = licensor(lic(LS, LN)) and subset(OS,operationset(lic(LS,LN))))
or iscreator(A,OS)).
--
ceq lic(restrict(LS, LN, A, OS), LN1)
  = (if not (LN1 = LN)
  then lic(LS, LN)
  else mk-lic(LN, A, licensee(lic(LS, LN)), del(OS, operationset(lic(LS, LN)))) fi) if c-restrict(LS, LN, A, OS).
ceq actor(restrict(LS, LN, A, OS), A1)
  = actor(LS, A1) if c-restrict(LS, LN, A, OS).
ceq docset(restrict(LS, LN, A, OS))
  = docset(LS) if c-restrict(LS, LN, A, OS).
ceq restrict(LS, LN, A, OS)
  = LS if not c-restrict(LS, LN, A, OS).

-- [4] withdraw
op c-withdraw : LicSys LicName ActorName -> Bool
eq c-withdraw(LS, LN, A)
  = (A = licensor(lic(LS, LN))) or iscreator(A, OS).
\[
\text{ceq lic}(\text{withdraw}(LS, LN, A), LN1) = (\text{if not } (LN1 = LN) \text{ then } \text{lic}(LS, LN1) \text{ else no-license }) \text{ if c-withdraw}(LS, LN, A).
\]
\[
\text{ceq actor}(\text{withdraw}(LS, LN, A), A1) = (\text{if } (A = A1) \text{ then } \text{mk-actor}(A, \text{del}(LN, \text{actorLicor}(\text{actor}(LS, A))), \text{actorLicee}(\text{actor}(LS, A))) \text{ else actor}(LS, A1) \text{ if c-withdraw}(LS, LN, A).
\]
\[
\text{ceq docset}(\text{withdraw}(LS, LN, A)) = \text{docset}(LS) \text{ if c-withdraw}(LS, LN, A).
\]
\[
\text{ceq withdraw}(LS, LN, A) = LS \text{ if not c-withdraw}(LS, LN, A).
\]

\[
\text{-- [5] run}
\]
\[
\text{op c-run : LicSys LicName ActorName Operation -> Bool}
\]
\[
\text{eq c-run}(LS, LN, A, O) = (\text{if } (\text{lic}(LS, LN) = \text{no-license}) \text{ then iscreate}(O) \text{ and } \text{(not } (\text{docName}(\text{doc}(O)) /\text{in docset}(LS))) \text{ else } (A = \text{licensee}(\text{lic}(LS, LN))) \text{ and } (O /\text{in operationset}(\text{lic}(LS, LN))) \text{ fi} .
\]

\[
\text{--}
\]
\[
\text{ceq lic}(\text{run}(LS, LN, A, O), LN1) = \text{lic}(LS, LN1) \text{ if c-run}(LS, LN, A, O).
\]
\[
\text{ceq actor}(\text{run}(LS, LN, A, O), A1) = \text{actor}(LS, A1) \text{ if c-run}(LS, LN, A, O).
\]
\[
\text{ceq docset}(\text{run}(LS, LN, A, O)) = (\text{if } \text{iscreate}(O) \text{ then add(docName}(\text{doc}(O)), \text{docset}(LS)) \text{ else docset}(LS) \text{ fi} ) \text{ if c-run}(LS, LN, A, O).
\]
\[
\text{ceq run}(LS, LN, A, O) = LS \text{ if not c-run}(LS, LN, A, O).
\]

\[
\text{open LICENSES}
\]
\[
\text{-- example 1:}
\]
\[
\text{-- in this example, we test in initial state, an actor "gf" play an}
\]
\[
\text{transition action "run"}
\]
\[
\text{-- which refersto create a document "af"}
\]
\[
\text{based on an arbitrary document}
\]
\[
\text{ops s s' : -> LicSys .}
\]
\[
\text{ops ln ln' : -> LicName .}
\]
\[
\text{ops gf ca : -> Actor .}
\]
\[
\text{ops af fb : -> DocName .}
\]
\[
\text{op os : -> OperationSet .}
\]
\[
\text{op as : -> ActorSet .}
\]
\[
\text{eq s =run(init,ln,gf,create(mk-doc(af,gf,none),arbitrary)) .}
\]
\[
\text{close}
\]
Bibliography


Publications

Conference papers


Technical Report
