ABSTRACT

UDUPI, YATHIRAJ BHAT. Modeling Service Engagements in Dynamic Organizations: Multiagent Model and Architecture for Policy-Based Governance. (Under the direction of Professor Munindar P. Singh.)

Service engagements arise commonly in business and scientific computing. A service engagement is characterized by autonomous parties coming together in a contractual arrangement to share resources or carry out tasks for one another. The autonomy of the participants is key, meaning that there is no unique locus for policy application. Yet, autonomy is not properly treated by current approaches for designing service engagements, which typically take the perspective of one of the participants.

We provide an agent-based conceptual model for specifying service engagements as arising within dynamic service organizations or Orgs. The atoms of a service engagement are formalized as commitments among the participants, to be created and manipulated as the engagement progresses. An Org scopes the commitments formed in an engagement. Orgs and their members are modeled as agents. A service contract provides a natural arms-length abstraction for modeling service engagements and is formalized as comprising a set of commitments among the contracting agents. An institution is a kind of an Org that acts as a social and legal context, within which Orgs arise modeling different service engagements.

We provide a policy-based architecture for the governance of service engagements. We propose innovations in policy-based architecture to model and govern complex Orgs. Traditional policy-based frameworks emphasize reactive behaviors wherein an external request causes a policy engine to compute a response. However, business service settings require richer policies and call for proactive behaviors. A business must not only respond to explicit requests, but also monitor its environment, collate events, and potentially act in anticipation of events in order to ensure that its policies are satisfied.

The core of this research is an approach to formalize service engagements based on commitments and to study their dynamics in the presence of policies specified by each of the participants. We describe a methodology for service engagement design using a set of design patterns that capture commonly occurring elements of service engagements. We have implemented MAVOS, a prototype of a policy-based multiagent system to model Orgs that demonstrates and evaluates our approach on realistic scenarios involving Orgs.
Modeling Service Engagements in Dynamic Organizations: Multiagent Model and Architecture for Policy-Based Governance

by
Yathiraj Bhat Udupi

A dissertation submitted to the Graduate Faculty of North Carolina State University in partial fulfillment of the requirements for the Degree of Doctor of Philosophy

Computer Science
Raleigh, North Carolina
2008

Approved By:

Dr. Gregory T. Byrd
Dr. James C. Lester
Dr. Ting Yu
Dr. Munindar P. Singh
Chair of Advisory Committee
DEDICATION

To Anu

for everything
Yathiraj B. Udupi hails from Udupi, a temple town in southern India, and a global trademark for vegetarian cuisine. He received his Bachelors in Technology in Computer Science and Engineering from the Indian Institute of Technology Madras (IIT-Madras), Chennai, India in May 2002. He received his M.S. in Computer Science from North Carolina State University in December 2005. He is pursuing his Ph.D. in Computer Science at North Carolina State University under the direction of Dr. Munindar P. Singh. During his graduate studies at NC State, he interned at Hewlett Packard Laboratories and IBM TJ Watson Research Labs. He taught a Computer Science course titled “E-Commerce Technologies” at NC State in the Fall of 2007, and had earlier been a teaching assistant for the courses “Web Services” and “Service-Oriented Computing” at NCSU.
ACKNOWLEDGMENTS

I thank my advisor Dr. Munindar P. Singh for his guidance, support, inspiration, and help I have received from my first day as a student at NC State. It has been a great rewarding learning curve in all aspects under his guidance. I thank my committee members, Drs. Gregory Byrd, James Lester, and Ting Yu, for their insightful comments. I also thank all my professors who enriched my learning experience at NC State.

My research internships have helped me get an industrial research perspective. I thank my mentors Drs. Steve Battle, Akhil Sahai from HP Labs, Michael Maximilien, and Stefan Tai from IBM Research Labs for having given me an opportunity to intern and having guided me through the internship.

I am indebted to my current and former colleagues at Multiagent Systems and SOC Laboratory, Dr. Zhengang Cheng, Amit Chopra, Dr. Nirmit Desai, Adel Elmessiry, Scott Gerard, Chung-Wei Hang, Chris Hazard, Prashant Kediyal, Dr. Ashok Mallya, Dr. Michael Maximilien, Derek Sollenberger, Pankaj Telang, and Dr. Pınar Yolum for their valuable intellectual company during my graduate studies and their feedback.

This dissertation was supported by research grants from NSF (grant ITR-0081742) and the Center for Advanced Computing and Communication (CACC), NCSU. My conference travels were supported by scholarships from AAAI, IFAAI, NCSU.

I thank all my friends at NC State who made my graduate life very fun filled. I thank Mithun Acharya, Kishan Dayal, Ravi Jenkal, Siddharth Savadatti, and Mitul Zaveri for being great roommates and for creating an amazing home away from home with delicious home made South Indian vegetarian food served every day.

I am indebted to Lalitha akka’s family and Chini didi’s family. These two families have been family away from home for me and their care and affection have kept my homesickness at bay.

I am forever grateful to my wife Anupama, for her love, encouragement, and patience through the busy days of working toward my dissertation and paper submissions. She has brought a new meaning to my life and I could not have come to this point without her support. I thank amma, appa, my sisters Sahana, and Vidya for believing in me and being a driving force all the way from my school days. Their blessings and good wishes mean a lot to me. I also thank my in-laws for all their well wishes and their belief in me.
# TABLE OF CONTENTS

## LIST OF FIGURES

<table>
<thead>
<tr>
<th>Figure</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>v</td>
<td></td>
</tr>
</tbody>
</table>

## LIST OF TABLES

<table>
<thead>
<tr>
<th>Table</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>vii</td>
<td></td>
</tr>
</tbody>
</table>

## 1 Introduction

<table>
<thead>
<tr>
<th>Section</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>1.1</td>
<td>1</td>
</tr>
<tr>
<td>1.1.1</td>
<td>1</td>
</tr>
<tr>
<td>1.2</td>
<td>4</td>
</tr>
<tr>
<td>1.2.1</td>
<td>4</td>
</tr>
<tr>
<td>1.2.2</td>
<td>5</td>
</tr>
<tr>
<td>1.2.3</td>
<td>6</td>
</tr>
<tr>
<td>1.3</td>
<td>7</td>
</tr>
<tr>
<td>1.4.1</td>
<td>9</td>
</tr>
<tr>
<td>1.4.2</td>
<td>9</td>
</tr>
<tr>
<td>1.5</td>
<td>11</td>
</tr>
<tr>
<td>1.6</td>
<td>12</td>
</tr>
</tbody>
</table>

## 2 An Agent-Based Conceptual Model

<table>
<thead>
<tr>
<th>Section</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>2.1</td>
<td>13</td>
</tr>
<tr>
<td>2.1.1</td>
<td>13</td>
</tr>
<tr>
<td>2.2</td>
<td>14</td>
</tr>
<tr>
<td>2.2.1</td>
<td>16</td>
</tr>
<tr>
<td>2.2.2</td>
<td>18</td>
</tr>
<tr>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
</tr>
<tr>
<td>2.2.3</td>
<td>22</td>
</tr>
<tr>
<td>2.2.4</td>
<td>23</td>
</tr>
<tr>
<td>2.2.5</td>
<td>24</td>
</tr>
<tr>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
</tr>
<tr>
<td>2.2.6</td>
<td>26</td>
</tr>
<tr>
<td>2.2.7</td>
<td>28</td>
</tr>
<tr>
<td>2.3</td>
<td>31</td>
</tr>
<tr>
<td>2.4</td>
<td>35</td>
</tr>
<tr>
<td>2.4.1</td>
<td>35</td>
</tr>
<tr>
<td>2.4.2</td>
<td>36</td>
</tr>
</tbody>
</table>
3 Organizational Design Patterns ........................................... 38
  3.1 Organizational Design Patterns ....................................... 39
    3.1.1 Example Org Patterns ........................................... 41
    3.1.2 An Example Service Engagement ................................ 49
  3.2 Formal Results ....................................................... 50
    3.2.1 Structure Based on Design Patterns ............................. 51
    3.2.2 Sound Service Engagements ..................................... 53
  3.3 Discussion ........................................................... 55
4 Policy-Based Governance in Orgs ........................................ 56
  4.1 Background: Traditional Policy Architectures ......................... 56
    4.1.1 IETF Policy Framework ........................................... 56
    4.1.2 XACML Policy Framework ........................................ 57
    4.1.3 Grid Policies Architectures ..................................... 58
  4.2 Org Policy Architecture Requirements ................................ 59
  4.3 Facets of a Policy .................................................... 59
  4.4 Agent Policy Architecture ............................................ 60
  4.5 Dynamics of Service Engagements arising in Orgs ...................... 62
    4.5.1 Creation of Orgs within Institutions ............................ 62
    4.5.2 Hierarchical Policy Enforcement within an Org ................ 64
    4.5.3 Policy-Based Control of Agent Actions .......................... 66
    4.5.4 Enactment of Commitments ....................................... 66
       Chain of commitments .............................................. 68
  4.5.5 Life Cycle of Orgs ................................................ 72
  4.6 Evaluation .......................................................... 73
    4.6.1 Challenges: Preemptive Scheduling .............................. 73
    4.6.2 Handling Conflicts and Escalations .............................. 75
    4.6.3 Formal Results on Chain of Commitments in Orgs ............... 76
    4.6.4 Formalizing Policy Compliance in an Org ........................ 79
    4.6.5 Point of Enactment ............................................. 79
    4.6.6 Delegation Path ................................................. 80
    4.6.7 Assignment Path ................................................. 80
    4.6.8 Contract Formation and Org Enactment ........................... 81
    4.6.9 Coalitions ........................................................ 82
    4.6.10 Actions and Policy Compliance .................................. 83
  4.7 Evaluation by Prototyping Practical Scenarios ......................... 87
    4.7.1 Grid Use Case Scenarios .......................................... 88
       Commercial Data Centers Use Case ................................... 88
       Severe Storm Modeling .............................................. 93
       Online Media and Entertainment .................................... 94
       National Fusion Collaboratory ...................................... 94
       Services-Based Distributed Query Processing ..................... 95
       Grid Workflow ....................................................... 95
Grid Resource Reseller ........................................... 96
Inter-Grid ............................................................. 96
Interactive Grid ...................................................... 96
Grid Lite .............................................................. 97
Persistent Archive .................................................. 97
Mutual Authorization .............................................. 97
Resource Usage Service .......................................... 98
4.8 Discussion ....................................................... 98
4.8.1 Detecting and Resolving Conflicts in Service Engagements ........................................ 99
4.8.2 Dependencies among Service Engagements .................................................. 99
4.8.3 Social Relationships Applied in Policies ................................................. 100
5 Related Work ....................................................... 102
5.1 Organizations and Multiagent Systems .................................................. 102
5.1.1 Organizations and Institutions ................................................. 103
5.2 Contracts and Organizations .......................................... 105
5.2.1 Electronic Contracts .................................................. 105
5.2.2 Contracts in Organizations ................................................. 106
5.3 Roles, Agents and Organizations .......................................... 107
5.3.1 Role-Based Organizational Design ................................................. 107
5.3.2 Role-Based Management Systems ................................................. 107
5.4 Policy Models, Languages and Architectures .......................................... 108
5.4.1 Policy Models ...................................................... 108
5.4.2 Policy Languages ...................................................... 109
5.4.3 Policy Architectures ...................................................... 110
5.4.4 Policy Architectures for Trust and Privacy Management ........................................ 111
5.5 Web Services Management Standards .......................................... 112
5.5.1 WS-Policy ...................................................... 113
5.5.2 WS-Agreement ...................................................... 113
6 Conclusion and Future Work .......................................... 114
Bibliography ...................................................... 117
LIST OF FIGURES

Figure 1.1 An example hierarchical scenario ........................................ 10
Figure 2.1 The agent-based conceptual model underlying our approach ....... 17
Figure 2.2 Delegations of commitments between DOE and DOC .............. 20
Figure 2.3 Assignments of commitments formed within Org-DOC-DOE ....... 21
Figure 2.4 An example scenario of Orgs and contracts .......................... 27
Figure 2.5 The conceptual model extended with institutions .................... 28
Figure 2.6 Example of auction Orgs in eBay .......................................... 30
Figure 2.7 Contract enactment in an Org ................................................ 34
Figure 3.1 Org pattern model ............................................................... 40
Figure 3.2 Delegation, assignment, and escalation patterns ....................... 42
Figure 3.3 Preemption and Reallocation patterns ................................... 45
Figure 3.4 An example service engagement scenario .............................. 49
Figure 3.5 Service engagement instantiations ....................................... 50
Figure 3.6 Org structure with role links ................................................. 51
Figure 4.1 The basic IETF policy framework .......................................... 57
Figure 4.2 A XACML policy framework ................................................ 58
Figure 4.3 Architecture of policy-driven agent ....................................... 61
Figure 4.4 The creation of Orgs ........................................................... 63
Figure 4.5 Hierarchical enforcement schematically ................................ 64
Figure 4.6 The life cycle of commitments .............................................. 69
Figure 4.7 Creating a chain of commitments ($C_1, C_2, C_3$) .................... 70
Figure 4.8 The life cycle of an Org ......................................................... 72
Figure 4.9 A preemptive scheduling scenario ....................................... 74
Figure 4.10 A messaging sequence diagram for the preemptive scheduling scenario 74
Figure 4.11 A conflicting commitments scenario .................................. 76
Figure 4.12 An example Org ................................................................. 79
Figure 4.13 Delegation and escalation patterns in the CDC use case scenario .. 92
Figure 4.14 Preemption and Reallocation applied to the CDC scenario ....... 93
Figure 4.15 An example scenario of contracts with dependencies .............. 100


**LIST OF TABLES**

<table>
<thead>
<tr>
<th>Table</th>
<th>Description</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>Table 2.1</td>
<td>Delegations of commitments between DOC and DOE</td>
<td>21</td>
</tr>
<tr>
<td>Table 2.2</td>
<td>Assignments of commitments</td>
<td>22</td>
</tr>
<tr>
<td>Table 2.3</td>
<td>An example institution specification</td>
<td>30</td>
</tr>
<tr>
<td>Table 3.1</td>
<td>Org pattern specification schema</td>
<td>41</td>
</tr>
<tr>
<td>Table 3.2</td>
<td>Delegation pattern specification</td>
<td>42</td>
</tr>
<tr>
<td>Table 3.3</td>
<td>Assignment pattern specification</td>
<td>43</td>
</tr>
<tr>
<td>Table 3.4</td>
<td>Escalation pattern specification</td>
<td>44</td>
</tr>
<tr>
<td>Table 3.5</td>
<td>Preemption pattern specification</td>
<td>46</td>
</tr>
<tr>
<td>Table 3.6</td>
<td>Reallocation pattern specification</td>
<td>47</td>
</tr>
<tr>
<td>Table 3.7</td>
<td>Collaboration pattern specification</td>
<td>48</td>
</tr>
<tr>
<td>Table 3.8</td>
<td>Representative pattern roles and role links</td>
<td>52</td>
</tr>
<tr>
<td>Table 4.1</td>
<td>Use cases and applicable Org patterns</td>
<td>98</td>
</tr>
</tbody>
</table>
Chapter 1

Introduction

The rising prominence of services poses new challenges to computing. The recent advances in services computing have made it easier for different autonomous and heterogeneous parties to transact with each other, build complex relationships, exchange services, and thus participate in complex *service engagements*. Current approaches and processes involved in handling these service engagements are complicated and often require hard coding or human intervention. Situations of exceptions and violations arising in such scenarios are handled on a case by case basis as they lack adequate modeling of the parties involved and their interactions. We emphasize the importance of modeling such service engagements and their enactment.

1.1 What are Service Engagements?

Service engagements are characterized by autonomous parties coming together in a contractual arrangement to share resources or carry out tasks for one another. They often involve the participants exhibiting complex behaviors by autonomously applying their own policies. The autonomous participants agree to collaborate and share resources, thereby administering themselves. Two main components of a service engagement are agreement and enactment.

Service engagements apply in business as well as scientific computing settings. In
scientific computing settings, service engagements are formed among entities such as individuals, enterprises, and owners of computational resources collaborating on some computational activity. These collaborations are created within virtual organizations (VOs) that aim to achieve the collective and individual goals of their members [Foster et al., 2001]. Traditionally such collaborative activities have focused on data sharing and computation. The grid computing technologies apply VOs to address the sharing of resources such as files, CPU cycles, software, data, and so on that are required by the various scientific, engineering, and industrial applications. In such computational environments, resource sharing rules or policies are specified by the various service providers and consumers.

In business settings service engagements are seen in cross-organizational service contracts, partnerships, supply chains, and so on. Such service engagements lead to complex relationships and multiple service agreements binding the participating entities. The participating entities create commitments toward each other that may be required by the service engagement. A service engagement is unlike a transaction that ends immediately, but lasts until all the participants complete their commitments.

1.1.1 Example Service Engagement Scenarios

We present some examples of how even simple service engagements can turn out to require complex behaviors. Consider a scenario involving service engagements occurring in an online marketplace such as eBay. eBay regulates how members can join eBay. eBay supports the creation of an indefinite number of auctions, bringing different buyers and sellers together. Consider an auction for an item created by a seller. Several bidders participate in the auction and when the auction ends, the highest bidder is chosen as the buyer. The buyer and the seller now enter into a service engagement, where the seller promises to ship the won auction item to the buyer once he receives the payment from the buyer. Although simple, the above is a useful example of a service engagement because it involves the parties committing to each other. Here an engagement may last a long time until the buyer receives the item and has no complaints about it. The service engagement is usually fulfilled outside eBay, as it normally involves other parties that are not necessarily members of eBay. The buyer would send the payment to the seller via a credit card or a payment service provider such as PayPal (which only incidentally happens to be associated with eBay). The seller would use one of the shipping service providers such as DHL. The service
engagement thus extends beyond the direct control of eBay. The key processes of a service engagement run across multiple organizations involving different administrative domains. Current approaches do not account for multiple autonomous parties and thus cannot model even such simple engagements adequately.

Service engagements are created in the area of provisioning of services, where sophisticated service level agreements are required to manage the resources and services controlled by multiple organizations. A production grid facility and a hurricane modeling facility may create a service engagement, where the former provides grid services to the latter. The grid facility might be involved in several such service engagements and hence could potentially fail to satisfy some agreements. It may provide preemptive priority to the hurricane modelers to react to emergent environmental situations and generate more accurate warnings. This can be made possible by continually monitoring emergent events and using policies that preempt the grid provider from other engagements. The above scenario might appear to be simple, yet it features as a challenge problem in grid computing [NGG2, 2004].

Consider a supply chain, where an organization might be involved in contractual agreements with different suppliers. Here, a supplier might have simultaneously created multiple contracts with different organizations, and sometimes might not be able to satisfy a contract, hence requiring complex mechanisms to resolve conflicts. Complex service engagements and challenging scenarios are possible in business process outsourcing (BPO), which involves refactoring an enterprise’s processes so that external parties are engaged, yielding improvements such as efficiency and reliability.

The service engagements often result in violations, delays, or cancellations. The parties involved should be able to seek recourse under such circumstances. For example, eBay provides a feedback and rating mechanism, where buyers or sellers can post any grievances about the auction they participated in. eBay may step in to determine if any member has to be penalized, based on the feedback received. The reason the above scenarios are challenging is because they involve more than one autonomous stakeholder, and there is not a unique point where a desired policy can be applied. Further the desired outcome depends upon how different business entities relate to one another. Anticipating the dynamics of such “configurations” when authoring policies is clearly not viable. Consequently current approaches mostly rely on hard-coded solutions or human intervention and do not adequately model the service engagements and their dynamics.
1.2 Motivation for Modeling Service Engagements

We model a service engagement as arising within a dynamic service organization, termed an Org here. An Org is a computational virtual organization that comes into existence when a group of autonomous parties come together in a contractual arrangement. An Org is created for a specific instance of an engagement. An Org created for a service engagement lasts only until the service engagement is fulfilled. In the eBay example discussed above, the service engagement resulting from the won auction can be modeled using an Org. Sellers and buyers are the autonomous parties coming together within this Org.

In the proposed approach, Orgs are considered at a higher level than the traditional approaches that provide the functionality (“physiology” [Foster et al., 2002]) and protocol-oriented descriptions (“anatomy” [Foster et al., 2001]) of virtual organizations (VOs) with respect to resource sharing and computations among them.

1.2.1 Traditional Approaches: VOs in Grid Computing

Foster et al. [2001] define the main and specific problem that underlies the Grid concept as a coordinated resource sharing and problem solving in dynamic, multi-institutional virtual organizations. They present a layered grid protocol architecture as an “hourglass model” consisting of Resource, Connectivity, Collective, and Application layers over the Fabric layer that provides the resources requiring shared access. The Resource and Connectivity protocols facilitate sharing of individual resources, while the Collective layer coordinates multiple resources. Application layer comprises the user applications that operate within a VO environment. Foster and colleagues [2002] address the requirement for an integrated service architecture across distributed, heterogeneous, and dynamic VOs and present an open grid service architecture. Here, they present a functional perspective for the grid problem addressing how the grid functions, and how the grid technologies can be implemented and applied.

Our approach proposes a multiagent architecture for Orgs and a proactive policy-based governance architecture. We study how the Orgs and their members are administered to ensure the proper enactment of the service engagements in the presence of the policies of the participants. The autonomy of the participants is key and decisions are made by peers. Hence, governance is a better metaphor for administering Orgs than traditional
management. Governance involves the processes by which autonomous entities agree to collaborate and share resources. We emphasize the need for better governance and modeling mechanisms for Orgs. The following subsections motivate our approach.

1.2.2 Why a Multiagent Architecture?

Service engagements and Orgs in business settings are characterized by processes that support the delivery of real-world (not just IT) services. Because of legal and economic pressures, business environments provide richer policies than the more common VO scenarios in scientific computing environments. Different business entities collaborate with each other, and enter into contracts (i.e., mutual agreements) by forming commitments (i.e., obligations) to achieve certain goals. These collaborations result in the formation of Orgs in a business environment. The following are key properties that distinguish Orgs from traditional IT architectures:

**Autonomy.** The members of an Org behave independently, constrained only by their contracts with each other.

**Heterogeneity.** The members of an Org are independently designed and constructed, their implementations being constrained only by the applicable interface descriptions that might be adopted.

**Membership Dynamism.** The members of an Org join and leave with minimal constraints. In effect, the configuration of an Org changes at runtime without requiring the Org to be stopped and restarted.

**Structural Dynamism.** Orgs have complex internal structures, reflected in the relationships among their members. These relationships evolve as business relationships are created, modified, and terminated.

The above properties closely match the properties of multiagent systems [Huhns and Singh, 1998b; Weiss, 1999]. Agents are persistent computations representing independent principals: they are autonomous and heterogeneous as a result. Multiagent systems are motivated from flexible human organizations and consequently exhibit membership and structural dynamism. The autonomous parties coming together in service engagements are best modeled as agents. An Org itself can be modeled as an agent, and it contains other member agents.
(individual entities or Orgs). An Org created for a service engagement lasts only until the service engagement is fulfilled. A service engagement is characterized by a set of related commitments created by the participating agents toward each other.

1.2.3 Policy-Based Governance Architecture for Orgs

Orgs consist of autonomous, heterogeneous, and dynamic members, often exhibiting complex behaviors and structures and hence governing these complex Orgs is an important concern. Orgs, being complex are a natural match for policy-based approaches. We seek to improve the modeling and governance of complex Orgs through innovations in policy architecture. Traditional policy-based frameworks emphasize reactive behaviors, wherein an external request causes a policy engine to compute a response. These reactive policy frameworks are adequate for applications such as routing packets or controlling access to data items.

However, business service settings require richer policies and call for proactive behaviors. For example, in scientific computing, an Org representing a “production grid” would deal with resources (e.g., instruments in labs or observatories), representing the real world. In order to ensure the satisfaction of its policies and contracts, such an Org must not only respond to explicit requests, but must also monitor its environment, collate events, determine compliance of its interactions, and potentially act in anticipation of events. For example, a grid facility may assign preemptive priority to hurricane modelers, to react to emergent environmental situations so as to generate more accurate warnings for the public. Preemptive scheduling, though apparently simple, remains a major challenge in grid computing [NGG2, 2004]. In a business supply chain scenario, companies would regularly monitor suppliers’ production to anticipate and work around any potential disruptions. Modern needs, such as resource sharing in cyberinfrastructures, and accommodating IT and other service engagements, speak to the importance of policy-based governance. Thus, a proactive policy-based governance architecture is naturally suited for the interoperability (here, we mean mutual interactions) of the members of the Orgs.
1.3 The Proposed Approach

We propose a multiagent conceptual model for Orgs, where agents are specified using a proactive policy-based architecture. Our proposed approach has the following key features.

**Contractual basis.** The key aspect of Orgs from our perspective is how they are characterized via relationships among their members. Specifically, these relationships are in the form of contracts. The proposed model supports the dynamic formation and enactment of contracts with a flavor of policies. A service engagement arises from a contract created among different agents. Two or more agents can create service engagements by sharing services, resources, and thereby accomplish certain individual or collective goals.

**Management of context.** The nested structure of Orgs facilitates each Org to provide the context for the interactions among and the policies of its members, both individual agents and other Orgs. All service engagements among the interacting agents are formed within the scope of their context Org. The context functions as a neutral organization that ensures the correct interactions of agents within and across organizations under different circumstances. The context thus provides the scope for the relationships, contracts, interactions among, and policies of its members. The key feature of our approach is that we support recursive contexts. An Org can form a context to other Orgs who themselves are contexts to their member agents.

**Institutional scoping.** To naturally model and enact real-life engagements, it is helpful to introduce the notion of an *institution*. An institution is an Org that has an identity of its own. Its life time may exceed that of its members. Universities such as NCSU and markets such as eBay are examples of institutions. Several service Orgs arise within institutions. An institution provides a (social or legal) environment for the dynamic service Orgs that come into existence within its scope. An institution provides the entry and exit policies and may specify certain restrictions on the service engagements occurring within it. For example, thesis committees are formed within the NCSU institution. In eBay, the Orgs representing won auctions exist temporarily within the eBay institution, and are subject to eBay’s requirements. Each won auction has exactly one (registered) seller and one highest (registered) bidder selected as the buyer. A business service contract created by an organization that is accredited with BBB
Commitments basis. *Commitments* naturally fit as the underlying representations for service engagements. Commitments are highly suited for modeling and enacting service engagements for the following two reasons. One, commitments capture the essence of obligations in a three-party manner, involving a debtor, a creditor, and a context. By contrast, traditional deontic concepts involve no more than two parties, and can capture two-party relationships, but cannot capture the fact that the relationships may occur within the scope of a context. An Org created for modeling service engagements naturally forms the context for the corresponding commitments. Two, commitments are captured as first-class entities and can be manipulated through operations such as delegate, assign, release, and so on. The commitment operations provide a basis for managing the relationships among the members of an Org in rigorous yet computationally natural manner.

In our approach, a contract created among different agents comprises a set of commitments created by the contracting agents toward each other. An agent can commit to another to provide a certain service, or agree to share certain resources. These commitments and hence the corresponding contract are formed in the context of a common Org.

Policy formulation, governance, and enactment. The proposed approach differs from the traditional work on policies by emphasizing not a policy engine, but a conceptual model in which policies can be perspicuously formulated, as well as an operational semantics for the governance and enactment of policies within nested Orgs.

Service engagements created among different agents arising within an institution are constrained by the policies of the institution. The policies of an institution would normally be inherited by the Orgs arising within them and hence apply as policies of the member agents. The policies control the agent actions and thus many decide how the commitments evolve. For example, an Org arising within an institution and hence inheriting the institution’s policies might declare a contract invalid or completed, or might release any of the associated commitments. This level of flexibility is essential
for an Org to handle exceptions and accommodate opportunities. Thus an Org can have policies, and an agent that is a member of an Org would normally be expected to satisfy any policies defined by the enclosing Org.

1.4 Example Scenarios of Orgs

The following subsections elaborate two example (fictitious, but realistic) scenarios – (1) a business supply chain scenario, and (2) a service provisioning contract scenario. Business scenarios such as these are two of the running examples used to describe the proposed approach in the later chapters.

1.4.1 A Business Supply Chain Scenario

Consider a scenario where a car manufacturer, Ford starts the process of manufacturing a new car. As a part of its supply chain, a tire manufacturer Goodyear provides the necessary tires. The deal is captured as a contract within the scope of a context Org that is created comprising the members Ford and Goodyear. The contract may specify a commitment from Goodyear to Ford to supply the adequate number of tires within a stipulated period of time (say, hundred thousand tires every year for five years), and another from Ford to Goodyear to make the payment on time and to buy tires only from Goodyear for five years. However, there can arise scenarios of potential disruption of production of tires by Goodyear. Ford can run into losses if it cannot do a timely release of its manufactured cars. Hence, there is a need for a mechanism that monitors the suppliers to anticipate and work around any potential disruption. The proposed approach addresses this challenge and provides a proactive policy-based governance architecture for the enactment of the above service engagements.

1.4.2 A Service Provisioning Scenario

Consider a scenario where the Department of Energy (DOE) and the Department of Commerce (DOC) coexist within an Org Org-DOC-DOE, which is created as a context for the service agreements between DOC and DOE. Figure 1.1 illustrates this scenario. The dashed
edges indicate the hierarchical organizational structure. DOC contains NOAA, which contains the National Hurricane Center (NHC). Likewise, DOE contains National Labs (NL), which contains Argonne (ANL) and Oak Ridge (ORNL). Here the leaf nodes, e.g., NHC and ANL, are individual agents; the other nodes are Orgs. Enactment can occur at multiple levels in the hierarchical structure. Hence, other service agreements or relationships can exist between members resulting in the formation of new Orgs. For example, Org-NOAA-NL is created as a context for NOAA and National Labs, and Org-NHC-ANL for NHC and ANL.

ANL, a descendant of DOE, provides grid computing facilities to the NHC, which performs hurricane modeling. ANL is committed to NHC to providing the computing service. NHC commits to ANL to ensure accurate scheduling and supervision of the grid jobs, and also to make timely payments for the services obtained.

**Challenge: Preemptive Scheduling**

A challenging use case here is that of preemptive scheduling of resources such as in the light of critical events. Say, certain events analyzed by NHC indicate an emergency hurricane situation causing a surge in NHC’s computational resource requirement beyond what is
currently being offered by ANL. In this scenario, NHC makes a request for additional resources in the Org-NHC-ANL context to ANL. In this scenario, it is likely that ANL might be involved in multiple commitments, and sometimes, might not be able to honor the commitment to NHC for hurricane modeling. In case the request cannot be granted, ANL or NHC may escalate it to Org-NHC-ANL. If NHC escalates, Org-NHC-ANL will check the request before deciding its action. If the request is legitimate and if ANL cannot grant it, then Org-NHC-ANL can inform National Labs of the situation by escalating it further to Org-NOAA-NL. In well-designed organizations, exceptional situations can be handled locally. Thus forwarding of escalations is rare, but can propagate up to the topmost Org. The higher Orgs enforce their policies preemptively. The higher Orgs may preempt some service agreements in favor of others. Here, Org-NOAA-NL would request National Labs to provide the additional resources to NHC. National Labs may cause ANL or another member such as ORNL to give up any other service engagements to support NHC fully instead.

This might appear to be quite easy. Yet, preemptive scheduling features as a challenge problem for production grid applications because current architectures and policy models do not support acquiring the right requirements or modeling and placing the desired policies correctly to produce desired behaviors without hardcoding or human intervention. The proposed approach develops the necessary architecture that would enable such proactive policy-based governance.

1.5 Contributions

The core of this research is an approach to formalize Orgs and institutions, based on commitments and study the dynamics of the enactment of service engagements in the presence of the policies specified by each of the participants. We have implemented a prototype called MAVOS, a policy-based multiagent system to model Orgs that demonstrates and evaluates our approach by realizing some use case scenarios of Orgs.

Our contributions can be summarized as follows.

1. We provide an agent-based conceptual model for specifying service engagements as arising within Orgs, which are hosted within institutions.

2. We describe a methodology and provide a set of design patterns for specifying service engagements that identify the key stakeholders in the form of roles, and their
relationships in the form of commitments.

3. We describe a proactive agent-based policy architecture that supports a hierarchical model of policy monitoring, compliance checking, and enforcement. We provide a formal analysis of the dynamics of service engagements in the presence of policies of each of the agents participating in a service engagement.

1.6 Organization

Chapter 2 describes an agent-based conceptual model for service engagements arising within Orgs. It formally presents the concepts required to model service engagements. Chapter 3 provides a methodology of using design patterns to specify different classes of service engagements that identify the participating stakeholders as roles. It describes a set of design patterns that are applicable for service engagements arising within Orgs. Chapter 4 discusses our proactive policy-based governance architecture for Orgs and describes mechanisms for policy-based control of agent actions. It formally studies the dynamic aspects of the enactment of service engagements and presents life cycle analyses of commitments and Orgs. It provides formal results on policy compliance and completion of Orgs. Chapter 5 studies relevant background literature and provides a comparative evaluation of the proposed approach. Chapter 6 concludes with a discussion of future work and a summary of contributions.
Chapter 2

An Agent-Based Conceptual Model

Agents are persistent computations representing autonomous and heterogeneous principals. Multiagent systems have been used to model many scenarios of collaboration and cooperation involving autonomous parties interacting with each other. Example scenarios include coalitions, teams, organizations, federations, societies, distributed systems, peer-to-peer systems, and so on. This chapter describes the application of agent-based architectures to model Orgs. Section 2.1 provides some background literature that forms the basis for the proposed approach. The proposed approach is described in Section 2.2 and it provides an agent-based conceptual model and a rich, domain-independent vocabulary specifying Orgs, their configuration facts, and the interactions applicable in the enactment of the service engagements formed in Orgs.

2.1 Background

The following subsection describes key background research for the proposed approach. Chapter 5 provides other related work.

2.1.1 SoCom: Spheres of Commitments

Social commitments have become an important abstraction in the science of multiagent systems. Commitments are obligations held by agents because of the promises they make
to fulfill certain goals or tasks. Singh introduced the concept of a sphere of commitment (SoCom) to describe a group of agents, and a SoCom is viewed in conjunction with its roles and their commitments [Singh, 1999]. A sphere of commitment hence encapsulates the promises and obligations the agents may have toward each other. A SoCom is conceptually viewed as the scope or context within which a commitment applies [Jain et al., 1999]. A SoCom is a multiagent system that the agents constitute, and it serves as the context for commitments among those agents. Here, a SoCom is modeled using a representative agent, and can potentially participate in larger SoComs. An abstract SoCom defines the structure of a SoCom using roles, instead of actual agents. A role is an abstraction that characterizes the functional behavior of an entity (actor) that adopts it. A role also specifies the rights and obligations of the actors in a specific context. An abstract SoCom specifies the commitments associated with a role, and the agents playing that role are expected to satisfy these commitments. A concrete SoCom is obtained by naming an abstract SoCom, and assigning agents to its roles. A SoCom manager is responsible for the instantiation of abstract SoComs. Hence it is responsible for the correct assignment of roles to different agents to avoid any conflicting roles in the SoCom. Jain et al. apply SoComs in a manufacturing scenario to model virtual enterprises that are composed of several autonomous business entities and are capable of making joint commitments to their common customers [Jain et al., 1999].

The proposed approach applies commitments to model the relationships among different entities taking part in an Org. It builds on the notion of SoComs to model Orgs, but with an additional emphasis on the complex nested architectures, structural and membership dynamisms, and the policy-based governance of commitments among different agents constituting the Org.

2.2 The Static Conceptual Model

We present a conceptual model for the static aspects of Orgs. Our approach is centered on the notion of agents. An agent is a computational entity with a persistent identity that is proactive and interactive. We conceptually model Orgs as multiagent systems, where each individual and Org is modeled (recursively) as an agent. That is, an Org is an agent that comprises other agents, in particular, other Orgs. An agent is thus either a simple individual agent or an Org agent. An Org corresponds to a dynamic service organization
that is created for enacting and governing the service engagements formed between two or more peer agents. The proposed approach models Orgs as arising within institutions. An institution is a kind of an Org with a longer life time than any of the Orgs arising within. Each such Org might itself be an institution and thus host additional Orgs to solve more specific problems for their members.

In our approach, an Org modeling a service engagement and a real-life physical organization are computationally modeled in the same way. Hence, a real-life organization is also represented as an Org. The main difference between an Org and a real-life organization is that an Org is created for one instance of a service engagement, while a physical organization can be involved in multiple service engagements. An Org exists as long as the service engagement exists, which may last for a few minutes or hours; and may continue to operate for as long as any human physical organization.

Each agent maintains a set of policies and configuration facts. Agents come together to accomplish goals in the presence of policies. Agents may collaborate by sharing goals, forming contracts, and creating commitments among one another. Configuration facts describe these structures in which an agent participates. They include information about their existing relationships such as commitments and contracts, organizational structure, power relationships, and so on. Agents interact by manipulating the commitments created among each other using different operations defined on the commitments. The key features of our multiagent conceptual model for Orgs can be summarized as follows:

**Recursive formulation.** The multiagent conceptual model handles recursively formulated Orgs. Orgs are agents and they comprise other agents who could be individual agents or Orgs themselves. Orgs themselves may be formed within institutions, which host various member agents and act as a context to the Orgs arising within them. The hierarchies that we see in real-life organizations are easily captured using this recursive formulation. Orgs can thus easily model real-life organizations with their member hierarchy.

**Autonomy and local policy compliance.** The model allows for agents supporting each Org’s autonomy. The individual policies of an agent reflect its autonomy and the architecture supports this local policy compliance.

**Contractual basis.** The relationships among the member agents of an Org are captured
in the form of contracts. The dynamism of these relationships is captured by continuously monitoring the enactment of the contracts. The multiagent model supports such dynamism.

**Management of context.** The model allows for the management of context, which is provided by the Org enclosing the contracts and commitments among the member agents. The context provides the scope for the relationships, contracts, interactions among, and policies of its members. The recursive formulation of Orgs enables an Org to be a context for a set of other Orgs who are themselves contexts for different contracts.

Figure 2.1 illustrates a static view of our conceptual model for capturing Orgs (as a special kind of agents), agents’ policies and goals, the commitments and contracts among agents, and the operations that characterize the interactions among the agents. Here, the notation is based on UML class diagrams; cardinalities are not shown to reduce clutter. This can be thought of as providing a rich, domain-independent vocabulary for specifying Orgs, their policies, and other configuration facts that include the possible interactions for the enactment of service engagements, including the creation of new Orgs and contracts, and the enactment of current contracts and commitments.

Intuitively, business partners may interact by forming contracts, fulfilling commitments, while complying to their policies. The following organizational representations formalize such concepts.

### 2.2.1 Commitments

The central primitive for expressing organizational interactions among agents is that of commitments [Singh, 1999]. Commitments are highly suited to model Orgs. Commitments capture the essence of obligations in a three-party manner, involving a debtor, a creditor, and an organization serving as the context for the commitment. In our approach, a commitment is a directed obligation from a debtor agent to a creditor agent, within the scope of a parent Org, which forms the context. Commitments specify a condition that the debtor is obliged to bring about. A commitment may realize the individual goals of the agents in the commitment, or the collective goals of the context Org. Commitments provide a natural basis for service contracts. The context of a commitment provides the means to handle
Figure 2.1: The agent-based conceptual model underlying our approach.
exceptions and opportunities by delimiting the scope of a commitment to its context Org. Thus the context provides an organizational basis for revoking or otherwise manipulating commitments.

**Definition 1** A commitment is defined as \( C(D, Cr, F, O, Id) \), where \( D \) is the debtor, \( Cr \) the creditor, \( O \) the context, \( F \) the discharge condition, and \( Id \) the Id of the commitment.

In essence, commitments reify aspects of agent interactions and enable interactions to be treated as first-class citizens in our representations. The motivation for the context of a commitment is to delimit the scope of a commitment, so as to enable the proper treatment of exceptions and opportunities. In particular, commitments in real life are not irrevocable. Often, an agent has no choice but to revoke a commitment because of problems that may be, for instance, physical (factory burned down), economic (oil prices shot up unexpectedly), or legal (cannot ship pharmaceuticals across national boundaries). The context of a commitment provides an organizational basis for revoking or otherwise manipulating commitments.

Castelfranchi [1995] has addressed the interplay between commitments and social structure. This work defines operations on commitments and groups. Commitments help in modeling how business interactions proceed, and they enable easy detection and accommodation of any business violations or exceptions [Yolum and Singh, 2002]. Verdicchio and Colombetti study commitments in an agent-based supply chain management scenario [2002]. They apply the multiagent paradigm to model a supply chain network, which is a network of autonomous business units engaged in procurement, manufacturing, and distribution of related products. Commitments in a supply chain network specify the social contract that exists between two companies reflecting their business relations and binds one agent to another to perform a certain action within a stipulated period of time.

### 2.2.2 Commitment Operations

Commitments function as first-class entities, and can be manipulated through several operations. Originally six operations were defined on commitments [Singh, 1999], but now we consider the following. We explain the dynamics of these operations and present life cycles of commitments later in Section 4.5.4.
• Create. A commitment is created by a debtor agent.

• Discharge. A debtor may discharge the commitment by bringing about the stated condition.

• Cancel. A debtor can cancel the commitment.

• Release. The context Org or the creditor of a commitment can release it, relieving the debtor.

• Delegate. A debtor of a commitment may delegate it to another agent: the outcome of the delegation is that the delegatee becomes the debtor of a new commitment with the same condition and creditor as the original commitment. The new commitment is formed within the context of a new Org that is in the context of the original context.

• Assign. Similarly a creditor of a commitment can assign it to another creditor.

• Escalate. Intuitively, this operation conveys the meaning of the reverse of delegate or assign operation. An escalate of a commitment is sent upwards from a debtor or creditor to the context Org, notifying the delegator (or assigner) any cancellation of the delegated (or assigned) commitment. In general, a commitment may be escalated indicating any violations. When there is no response from the debtor of a commitment resulting in a time out, it is an implied cancel.

The operations discussed above are different ways of manipulating the commitments held by agents. These operations provide a computational basis for managing the relationships among the members of an Org. We elaborate on these commitment operations and illustrate their applicability in an Org scenario using examples.

Delegation

Delegation is an especially important concept while dealing with organizations. Castelfranchi and Falcone [1998] present a theory of delegation in cooperative agent-based systems, where delegation is characterized by a set of mental states (beliefs, intentions, goals, and so on.) of the agent involved in the interaction. A delegator is trying to achieve some of its goals through the delegatee’s actions. The authors also study how the different levels of delegation among agents decides their levels of autonomy. Pacheco and Santos study
delegations in role-based organizations [Pacheco and Santos, 2004]. The authors consider dynamic multiagent organizations with agents adopting different roles. Delegation refers to the interactions among agents, where we observe dynamic transference of obligations and permissions from one agent to another.

In our approach, in an Org scenario, the actions required for the successful discharge of commitments among agents within an Org, may be carried out either directly by the debtors or by a series of delegations. Debtors may delegate their commitments to other agents. The outcome of a delegation is that the delegatee ends up with the commitment and becomes the new debtor. However, we can leave a “trace” of the commitment with the delegator to handle any escalations. The new commitment is created within the context of a new Org created within the original Org (explained in detail in Section 4.5.4).

![Figure 2.2: Delegations of commitments between DOE and DOC](image)

In service contract scenarios, delegation is routine. For example, Figure 2.2 illustrates the delegations in the scenario described in Section 1.4.2. A dashed line directed from a debtor to a creditor indicates a commitment. DOC delegates its commitment $C_1$ (to make timely payments and to schedule and supervise the grid provisioning service) to NOAA, which delegates the resulting new commitment ($C_2$) to NHC (results in $C_3$). DOE delegates the commitment $C'_1$ of providing the grid service to NL, which delegates the resulting new commitment ($C'_2$) to ANL (results in $C'_3$). These delegations result in the creation of new commitments with the delegatees as the debtors.

Table 2.1 describes the above commitments and their delegations. The context Orgs of the commitments are not shown in the Figure 2.2. Here, $Org-DOC-DOE$ is the context Org for the commitment $C_1$. Delegations result in other sub-Orgs such as $Org-$
Table 2.1: Delegations of commitments between DOC and DOE

The commitment $C_1$ from DOC to DOE and its delegations are represented below:

$C_1 := C(DOC, DOE, Org-DOC-DOE, \phi_1, Id-1)$

$\text{delegate}(C_1, DOC, NOAA) \Rightarrow C(NOAA, DOE, Org-NOAA-DOE, \phi_1, Id-2)$ is the delegated commitment $C_2$

$\text{delegate}(C_2, NOAA, NHC) \Rightarrow C(NHC, Org-NHC-DOE, \phi_1, Id-3)$ is the delegated commitment $C_3$

The commitment $C'_1$ from DOE to DOC and its delegations are represented as below:

$C'_1 := C(DOE, DOC, Org-DOC-DOE, \phi_2, Id-1')$

$\text{delegate}(C'_1, DOE, NL) \Rightarrow C'_2 := C(NL, DOC, Org-NL-DOC, \phi_2, Id-2')$

$\text{delegate}(C'_2, NL, ANL) \Rightarrow C'_3 := C(ANL, DOC, Org-ANL-DOC, \phi_2, Id-3')$

$NOAA-DOE, Org-NHC-DOE, Org-NL-DOC,$ and $Org-ANL-DOC$. $\phi_1$ is the discharge condition for $C_1$, which holds true when the grid service provisioning job is correctly scheduled and supervised, and the payment is made, thereby discharging the the commitment $C_1$. $\phi_2$ is the discharge condition of $C'_1$ which means the grid service provisioning job is complete.

Figure 2.3: Assignments of commitments formed within $Org-DOC-DOE$

Other operations

**Assign operation.** The creditor of a commitment may assign its commitment to another agent. In the example above, after all the delegations, consider the commitment $C_3$, with
Table 2.2: Assignments of commitments

<table>
<thead>
<tr>
<th>Assignment</th>
<th>Commitment</th>
</tr>
</thead>
<tbody>
<tr>
<td>$\text{assign}(C_3, \text{DOE}, \text{NL}) \Rightarrow C_4: C(\text{NHC}, \text{NL}, \text{Org-NHC-NL}, \phi_1, \text{Id-4})$</td>
<td>$\text{assign}(C_4, \text{NL}, \text{ANL}) \Rightarrow C_5: C(\text{NHC}, \text{ANL}, \text{Org-NHC-ANL}, \phi_1, \text{Id-5})$</td>
</tr>
<tr>
<td>$\text{assign}(C'_3, \text{DOC}, \text{NOAA}) \Rightarrow C'_4: C(\text{ANL}, \text{NOAA}, \text{Org-ANL-NOAA}, \phi_2, \text{Id-4'})$</td>
<td>$\text{assign}(C'_4, \text{NOAA}, \text{NHC}) \Rightarrow C'_5: C(\text{ANL}, \text{NHC}, \text{Org-ANL-NHC}, \phi_2, \text{Id-5'})$</td>
</tr>
</tbody>
</table>

NHC as the debtor and DOE as the creditor, and the commitment $C_3$, with ANL as the debtor and DOC as the creditor. Figure 2.3 illustrates the assign operations in this scenario. DOC may assign the commitment $C'_3$ to NOAA, which results in the new commitment $C'_4$. NOAA may assign $C'_4$ to NHC, which results in $C'_5$. DOE may assign $C_3$ to NL, resulting in $C_4$, and NL may assign $C_4$ to ANL resulting in $C_5$. Table 2.2 illustrates the above assign operations.

**Cancel.** If ANL realizes that it cannot satisfy its commitment, it can cancel its commitment.

**Escalation.** Any cancellation or other violation of commitments result in escalations of commitments. Here, when the NHC finds out about the cancellation of the commitment by ANL, or any violation in the grid service provisioning job, it can escalate the commitment to the context Org.

**Release.** DOC being the creditor can release DOE from its commitment at any time. The context Orgs of these commitments can release any of the commitments. The Orgs would release the commitments, if they are violating any of the Org policies, for example, if it detects any illegal commitment. This way, the creditor of a commitment or the context Org may release the commitment.

**Discharge.** The debtor ANL discharges commitment $C'_5$ when $\phi_2$ holds, and similarly NHC discharges commitment $C_5$ when $\phi_1$ holds.

### 2.2.3 Goals

The goals of an agent capture the states of the world that the agent desires to bring about. Goals are most naturally thought of as ends, but they can readily function as means to
other goals. In connection with Orgs, the goals (ends) of some agents may cause them to enter into a contract or form an Org. Conversely, the contracts that an Org enters into may cause it to adopt goals (means), which could potentially yield additional goals for its children and other descendants. For example, in our supply chain scenario described in Section 1.4.1, Ford can have a goal of manufacturing twenty five thousand cars in a year. In order to achieve this goal (ends), one of the product requirements is the tire. Hence it enters into a contract with Goodyear, as we saw earlier. Goodyear, when it enters into a contract with Ford, may adopt a goal (means) to produce at least hundred thousand tires every year. The goals of an agent are realized by the actions undertaken by itself or by other contracting agents. Policies control the actions and hence control the accomplishment of goals.

To accomplish goals, agents create commitments among each other and hence the goals of an agent can be translated to the discharge conditions of the commitments. The actions undertaken by the agent with a goal, or by other contracting agents may discharge these commitments. Thus, a single goal of an agent may be accomplished by one or more commitments. A goal formally can be expressed as follows:

**Definition 2** The goal $G_{A_i}$ of an agent $A_i$ is represented as the tuple $(A_i, Conditions, state)$, where $Conditions = \{\phi_1, \ldots, \phi_n\}$, is the set of discharge conditions of the relevant commitments with $A_i$ as the debtor, and $state$ is done, active, or failed.

### 2.2.4 Policies

Policies are formal expressions that are expressed as rules to control the actions undertaken by an agent. The actions are either domain actions or communicative actions (defined in Section 4.5.3). The communicative actions specify the various operations such as commitment operations, contract operations, administrative operations, and Org operations. Agent interactions are governed by its policies. The various operations are thus controlled by the policies of an agent.

The proposed policy architecture (discussed in detail in Chapter 4) incorporates the notions of monitoring and compliance checking for policies as first-class entities. The autonomy of each agent is preserved, by each agent having its own policy engine that
actively engages in monitoring and compliance checking. The policies of an Org agent are enforced locally by monitoring for any violations, and a distributed policy enforcement happens with the collaboration of the Org’s members, who inherit some of the parent Orgs policies. Policies may sometimes be adopted by agents (individual or Org agents) while entering into contracts with each other. For example, an Org agent may have a policy of invalidating a contract among its member agents, if the contract violates the Org’s policies. In our supply chain example, Ford may adopt a policy of not making commitments with any other tire manufacturer for five years, while it enters into a contract with Goodyear. In the DOC-DOE example, NL may have a policy to delegate all hurricane modeling grid service provisioning commitments to ORNL.

Policies are expressed as declarative rules. Each agent has a knowledge base, which includes a set of facts and the policy rules. In the implementation of our policy architecture, we use the Jess rule language and the Jess rule engine to express and run the policies.

**Definition 3** A policy is a rule that can be expressed as following:

\[ LHS \Rightarrow RHS \]

Here, \( LHS \) is a set of facts that can be combined using the operators and, or, or not. \( RHS \) is a set of function calls. The above rule implies that if the \( LHS \) holds true, then the policy engine runs all the function calls. The function calls may result in any assertions of new facts or any actions undertaken by the agent.

This way, the actions of an agent are controlled by the policies, hence policies govern the accomplishment of agents’ goals.

### 2.2.5 Contracts

A service contract among two or more agents encapsulates a related set of commitments. Like a commitment, a contract exists within a context Org. Each contract can be thought of as having its own unique context Org, although the context Org would occur within the scopes of existing Orgs. Moreover, the context Org of a contract would itself be or would enclose the contexts Orgs of the commitments that constitute the contract. This is
important, because during enactment, the commitments of a contract could be manipulated in various ways and, though they may end up with different contexts, such contexts would remain within the scope of the context of the given contract. If two agents create a contract with commitments existing toward each other, then we consider just one context Org for these commitments, which also forms the context Org of the contract.

When agents enter into a contract, they agree to the terms and conditions of the contract. These terms and conditions translate to the individual commitments of the contracting agents. Some of the terms and conditions may also translate to several policies that are adopted by the agents.

**Definition 4** A contract is defined as $Cn = (M', S', P')$, where $M' = \{A_1, \ldots, A_n\}$ is a set of contracting agents, $S'$ is a set of commitments, and $P'$ is a set of policies. Each commitment in $S'$ involves a debtor and a creditor drawn from $M'$.

**Contract operations**

The operations on a contract can have ramifications on the commitments formed within this contract, and vice versa. We define the following important operations on contracts.

- **Create.** A contract is created when a set of agents collaborate to accomplish their individual or collective goals by forming commitments among each other.

- **Complete.** A contract can be completed by the context Org if all the commitments specified in the contract are successfully discharged.

- **Release.** A contract can be released by the context Org, which essentially eliminates all the commitments. A context Org can also release the commitments of an individual contracting agent, in effect releasing him from the contract.

- **Cancel.** A contract is canceled by the context Org if the commitments fail to discharge or if they cause violations. This avoids nontermination and can result in the formation of a new contract.

Interesting scenarios arise when an agent is involved in two or more contracts simultaneously. The Orgs within which the contracts are formed may overlap, with the
agent belonging to different Orgs. An Org may release the commitments of the agent from one contract in order for the agent to be involved in a second contract. The structures of the Orgs and the different ways of contractual binding of its descendants can have various implications on enactment of contracts.

2.2.6 Orgs

A dynamic service organization or an Org is an agent that comprises other agents, in particular other Orgs. That is, an agent may be either an individual (representing a person, business partner, or a resource), or an Org. An Org is dynamically created to form the context for the service contracts and commitments formed among two or more peer agents. Please note that an Org in this sense is a (virtual) computational entity. Real-life organizations can be modeled as Orgs. We can create as many Orgs as needed. An Org’s member, their members, and so on, are called its descendants.

Definition 5 An Org $A_O$ is an agent that encloses a set of zero or more member agents $(M = \{A_1 \ldots A_n\})$.

An Org may include a set of organizational policies $P$ and a set of goals $G$. It forms the context for a set of commitments $S$, each of which has a creditor and a debtor drawn from $M$. It may also include other configuration facts. The Org may have its own goals, and may also adopt the goals from its member agents and their contracts. An individual agent is characterized by its goals, policies, and commitments.

Definition 6 An individual agent $A_I$ specifies a set of policies $P$, goals $G$, and includes a set of commitments $S$, each of which includes $A_I$ as either the debtor, or the creditor.

Each agent monitors and records events, which correspond to environmental observations or the actions of this and other agents. Based on these events and their policies, agents modify the configuration and choose their actions. The policies of an agent control its actions. An Org agent may specify certain policies that are inherited by its member agents. Policies play a key role in the enactment of service engagements and become important in administering Orgs. For example, an eBay seller agent may apply a shipping policy
to decide how to enact its shipping commitment to the buyer. For example, a shipping policy might determine whether to delegate the shipping commitment to FedEx or UPS. An auction Org in eBay might include policies (inherited from eBay’s policies) that restrict kinds of items that can be sold on eBay.

![Figure 2.4: An example scenario of Orgs and contracts](image)

We introduce some terminology using Figure 2.4. The root node is the Org X, with member agents A and B. Double arrow edges such as the one between A and B indicate a contract (could potentially be a multiparty contract). Single dashed edges drawn from top to bottom indicate a parent–child relationship. For example, the edge from A to A1 indicates that A1 is a child of A. The leaves E, F, G, H, I, and J are individual agents. Nonleaf nodes such as A, B, A1, A2, and X are agents representing Orgs.

As described earlier, a contract is formed within the context of an Org that encloses the contracting members. This Org is either formed on demand with the creation of a new contract, or a contract is formed within the members of an already existing Org. In Figure 2.4, a contract C₁ is formed between A and B in the Org X represented by their parent. Here, A either discharges its commitment to B directly, or may delegate it to any member Ai. Each delegated Ai should now discharge the delegated commitment. Here in the nested Org structure, the Org A can release A1 or A2 from any of their commitments that have A as their context (for example, commitments resulting from the contract C₂ between A1 and A2 in Figure 2.4).
2.2.7 Institutions

Institutions provide a means of describing the social environment within which Orgs (managing various service contracts) are hosted, i.e., come into being. An *institution* is a kind of an Org acting as a context to the Orgs it encloses. Each such Org might potentially host many additional Orgs to solve more specific problems for their members.

Figure 2.5: The conceptual model extended with institutions. Policies, goals, communicative actions, and contracts are not repeated here.

Figure 2.5 extends the conceptual model described in Figure 2.1 by modeling Orgs as arising within *institutions*.

**Definition 7** An *institution* (*I*) is a kind of an Org. An institution may enclose zero or more member agents (*M*), one or more *Org templates*, and forms the context for zero or more (instantiated) Orgs. It includes a set of *institutional* policies (*P*).

The policies of an institution address the administration of an institution but not the (domain-level) operations of its members. The policies determine agents joining and exiting the institution, the life cycles of the Orgs in the institution, their functioning, penalties, rewards, and so on. For example, eBay’s membership policies are entry and exit policies.

An *Org template* is defined below.
Definition 8  An Org template is represented by the tuple \( (R, Q, P_p) \). It encapsulates a set of member roles \( R \), a set of qualifications \( Q \), which specify constraints on the roles and include the relevant commitments, and a set of policy points \( P_p \), indicating where and what kinds of policies relevant to the specific commitment scenario being modeled apply.

Org templates are designed for an institution to support classes of service engagements with associated commitments formed among the members of an institution. Org templates are stored as templates in repositories and may be instantiated as needed. For example, eBay acting as an institution may define how a won auction may be settled using an Org template that specifies: the roles in such a transaction, constraints on such roles in terms of the applicable commitments, and a set of policy points, indicating the kinds of policies that apply for specific roles. The actual policies are authored by the agents adopting the roles when an instantiated Org is created. This way an institution provides certain institutional policies and Org templates for creating service contracts, but the agents have the autonomy to specify the actual policies of the contract.

Agents in an institution choose an Org template and instantiate the roles specified, which results in the creation of an Org. Within an institution, an Org template may be instantiated by different sets of agents to form separate Orgs. For example, an Org for the service engagement resulting from a won auction is created in eBay when a suitable auction Org template is instantiated. Here, the seller and buyer roles are instantiated by the agents by participating in the auction. In the eBay institution, several auction Orgs may exist simultaneously.

Exogenous Orgs.  Agents outside the institution may be involved in the enactment of the associated commitments. For example, a won auction arising within the eBay institution is fulfilled outside eBay when an auction item is shipped to a buyer after the seller receives the payment. Such Orgs with some members within and some members outside an institution are called exogenous Orgs of that institution.

Let us apply the above ideas to model a simplified online Marketplace \( M \) as an institution. Table 2.3 shows how \( M \) includes an Org template for the settlement of a won auction. Here, the example policies are stated in English, but can be easily implemented using a rule-based policy language such as Jess.

Figure 2.6 illustrates an enactment scenario based on this example. Solid arrows
Table 2.3: An example institution specification

<table>
<thead>
<tr>
<th>Institution Name:</th>
<th>M</th>
</tr>
</thead>
</table>
| Policies:         | Example policies of M include:  
P1: (Entry policy:) Members need to be at least 18 years of age and have a valid email address  
P2: (Exit policy:) A seller not paying the listing fee may be suspended from eBay |
| Org templates:   | An example Org template Θ specified in M for the settlement of a won auction is described below:  
| Roles:           | Seller, Buyer |
| Qualifications:  | Example qualifications are listed below:  
Q1: C_{S}: C(Seller, Buyer, \phi_S, \Theta) exists, where \phi_S means the buyer receives the item  
Q2: C_{P}: C(Buyer, Seller, \phi_P, \Theta) exists, where \phi_P means the seller receives the payment |
| Policy Points:   | Examples include:  
Pp1: Seller may specify a shipping delegation policy to delegate C_{S} to a shipper  
Pp2: Buyer may specify a policy for delegating C_{P} to a payment service provider |
| Member Agents:   | ABC, XYZ, Alice, Bob, Charlie. |
| Instantiated Orgs: | The instantiated Org O_{1} in the institution M is created with the role instantiations: Seller: ABC and Buyer: Alice |

![Figure 2.6: Example of auction Orgs in eBay](image-url)
indicate the delegations of the service commitments. Dashed arrows indicate the commitments with the arrow pointing from the debtor to the creditor. An oval indicates an Org. Agents Alice, Bob, Charlie, ABC, and XYZ meet the entry policies and join \( M \). Say, Alice wins an auction created by the seller ABC and instantiates the role of a buyer. Alice and ABC create commitments toward each other by instantiating the roles of the Org template \( \Theta \) and create an instantiated Org \( O_1 \). Here, Alice commits \( (C_P) \) to make the bid payment to ABC and ABC commits \( (C_S) \) to ship the auction item to Alice, both within a time specified in the contract. Alice makes the payment via PayPal by delegating \( C_P \) to PayPal, which creates a commitment \( D(C_P) \) to ABC within the context of an exogenous Org \( O_P \). ABC ships the book using DHL by delegating \( C_S \) to DHL, which creates a commitment \( D(C_S) \) to Alice within another exogenous context Org \( O_D \).

Service Orgs are formed within this institution among the members when auctions created by the sellers are won by the buyers. In simple terms, this means that agents must first find their way into an institution. Once there, they have established the credentials needed to create service engagements with their peers, the institution provides the necessary policies and the contexts for the enactment of the commitments formed among the agents.

2.3 Contract Enactment in Orgs

Our approach models hierarchical Orgs that arise within institutions when autonomous parties come together and create contracts. Our approach also supports nonhierarchical Org scenarios (such as exogenous Orgs where some parties outside of the institution are involved due to a series of delegations or assigns of the commitments formed within the institution). This section provides an analytical study involving the handling of exceptions and conflicts in contracts. Orgs can have complex nested structures and hence contracts may be formed at multiple levels. More than one contract may simultaneously exist among a set of contracting agents. Here, the Orgs within which the contracts are formed may overlap resulting in a situation where at least one agent belongs to more than one Org. We identify several different Org structures and their implications on contract enactment and vice versa. We identify some important cases based on the structure of the Orgs and the contracts formed at different levels. We show the effects of applying certain constraints or requirements on the behavior of the contracting agents. These cases mostly satisfy the
requirements enumerated below. We study these cases with the help of the illustrations in Figure 2.7.

**Requirements of Agents in Hierarchical Orgs.** The requirements and constraints specified below are applicable in hierarchical Orgs for agents forming contracts and Orgs using our approach. Some of these requirements are introduced as a mere technical convenience so that their implications can be studied. However, they may not be hard requirements. Hence, these requirements need not all hold simultaneously in some scenarios of Org enactment.

**R 1** An agent is restricted to communicate with its children, parents, and siblings.

An Org can handle exceptions and check for policy compliance among its children. However, the children should collaborate with their parents for enforcing the public policies of their parents. This limits the visibility of an agent to the outside world and any interaction is through its parent Org.

**R 2** Each of the commitments created within a single contract have the same context Org, which is also the context of the contract.

This requirement holds for the commitments originally formed for the contract. Any commitments created by delegation or assign of the original commitments have new Orgs created as their contexts.

**R 3** An agent has a dynamic behavior and can be involved in multiple contracts simultaneously. However, for each contract entered by an agent, there is a different Org.

This requirement allows an agent to be autonomous in deciding what contracts it wants to be involved with. An Org cannot have two contracts with a common contracting member in both the contracts. This requirement restricting one Org per contract is introduced as a technical convenience because it simplifies in the governance of the contract. However, any delegations or assigns inside a contract can be handled using many more Orgs.

**R 4** An Org agent can delegate its commitments only to its member agents.
This requirement enables distributed enactment of contracts. However, an Org may delegate its commitments only to member agents.

**R 5** An Org agent can assign its commitments resulting from a contract only to its member agents.

**R 6** An Org agent can release commitments that are formed either by a contract entered into only by its children, or by a delegation.

This requirement describes how a context Org can exercise the release operation over its children’s contracts and the associated commitments. For example, in Figure 2.4, A can release A1 or A2 only from the commitments of the contract C2, and not from any other contract formed between the children of B.

**R 7** An agent can escalate a complaint regarding a commitment or a contract only to its parent Org.

This requirement enables agents to administer their Orgs, contracts, and commitments by allowing for an escalate message to propagate up in the Org structure from a child to a parent.

**Exception handling in contracts.** An exception occurs when one or more of the parties involved do not behave according to the contract terms. Conflicting contracts cause exceptions. Exceptions are caught by the parent Orgs of the children involved in the responsible contracts when one or more of the contracting children send an escalation to the parent Org. For example, in Figure 2.7(b), if an exception is created in contract C3 between A and B resulting in an escalate to the parent Org Z, Z handles this escalate either by notifying the responsible party (A or B) about the exception and asking it to rectify it, or by releasing the agent causing an exception from the contract.

**Case 1:** In Figure 2.7(a), a contract C1 exists between A and B. Here, A can delegate its commitments from the contract to child A1. If A1 were previously committed to another
child of \( A \), say \( A_2 \), then \( A \) can release \( A_1 \) from that commitment. Applicable requirements: R 1, R 4, and R 6.

**Case 2:** Figure 2.7(b) shows a situation of overlapping Orgs with one Org having a contract \( C_3 \) between \( A \) and \( B \), and another Org having a contract \( C_4 \) between \( B \) and \( D \). Under these circumstances, \( A \) is not in the context of the \( B \leftrightarrow D \) contract that exists in a different Org. \( B \) has the choice to decide which contract to commit to at a particular time. \( B \) may be involved in both contracts if they do not conflict. Applicable requirements: R 1 and R 3. R 6 and R 7 become applicable if \( A \) requests \( B \) to commit to contract \( C_3 \), and \( B \) in the case of a conflict, escalates a message to its Org \( Y \), who can release \( B \) from the contract \( C_4 \).

**Case 3:** Figure 2.7(c) modifies Figure 2.7(b) to include a common ancestor Org \( X \) of \( A \), \( B \), and \( D \). Here, \( A \) is not in the context of the \( B \leftrightarrow D \) contract \( C_6 \), but \( A \) can escalate a request via its parent \( Z \) to \( X \). \( B \) and \( D \) are in the scope of \( X \) through their parent \( Y \). \( X \) can tell \( Y \) to release \( B \) from contract \( C_6 \). Applicable requirements: R 1, R 3, R 6, and R 7.

**Case 4:** In Figure 2.7(d), a three-party contract \( C_7 \) is formed between \( A \), \( B \), and \( D \).
Here, in the case of any conflicts, one of $A$, $B$, or $D$ can escalate a request to the parent $X$ who is the context Org of $C_7$. Applicable requirements: R 1, R 6, and R 7.

**Case 5:** In Figure 2.7(e), a contract $C_8$ exists between $B$ and $D$, with $A$ being a child of the same Org as $B$ and $D$. Here, $A$ cannot influence $B$ or $D$ in the $B \leftrightarrow D$ contract, but $A$ can escalate a request to its parent $X$ if it wants to contract with $B$ or $D$, and $X$ can release $B$ or $D$ from the contract. This is different from Case 2, but is like Case 3, because here $A$, $B$, and $D$ all have a common parent $X$. Applicable requirements: R 1, R 6, and R 7.

**Case 6:** In Figure 2.7(f) illustrates a scenario where $A$ and $B$ have two contracts between them for different goals. This results in the formation of two different Orgs $Z$ and $Y$ for the contracts $C_A$ and $C_B$, respectively. Under these conditions, conflicts may occur in deciding which Org has more influence over $A$ and $B$. These conflicts are resolved either by merging $Z$ and $Y$ (merging the contracts as well), or by making one of them supersede the other based on the contract precedences set by $A$ and $B$. Applicable requirements: R 1, R 3, R 6, and R 7.

### 2.4 Discussion

This chapter described our agent-based conceptual model and demonstrated the modeling of service engagements created among autonomous parties. We conclude this chapter with a discussion on nested institutions and Orgs and on overlapping Orgs.

#### 2.4.1 Analyzing Nested Institutions and Orgs

Institutions are kinds of Orgs. An institution has a life independent of its members; in particular, an institution may have a longer life time than its members. Our conceptual model supports nested institutions and Orgs. An institution can have member Orgs that are institutions themselves with their own entry, exit, and other institutional policies. Agents (individual agents or Orgs) who are members of an institution, can express willingness to join any member institution of the parent institution. A service engagement created within an institution between two member agents has a context Org that is created within this parent institution. The member relationship between an institution and another agent is also
modeled as a service engagement with a contract that lasts until the membership expires (either by eviction or by a mutually agreed upon contract end date). This membership contract has a context Org that is created within the parent institution. However, in the case of a topmost institution, we have an exceptional case, where any membership contract with an agent has its context Org as the topmost institution itself.

Dissolution of an Org may happen when all of its contracts and their associated commitments are in a done state. However, an institution may continue to exist until it is dissolved, even if it has no contracts or instantiated Orgs. The institution may dissolve an instantiated Org that is formed within it if any policies are violated. However the dissolution of an institution may be due to factors such as change or inapplicability of its policies. Agents may not be able to (or prefer to) join an institution if its policies are not applicable. For example, hypothetically, if an institution such as eBay has extremely strict membership exit policies and suspends members easily, fewer agents may prefer eBay. In the long run, this may result in the dissolution of eBay. The dissolution of an Org or an institution results in the dissolution of all the nested Orgs or institutions.

2.4.2 Overlapping Orgs

Our contract-based architecture supports overlapping Orgs. An agent can simultaneously be involved in multiple service contracts, and hence participate in multiple Orgs. Such a case suggests overlapping Orgs. The dynamics of enactment of commitments depend on the overlap and structure of the Orgs. Discussions below present some important scenarios of enactment in overlapping Orgs that are considered as future work.

Conflicting commitments. In overlapping Orgs, delegated or assigned commitments always start active, if the commitments formed in both the Orgs are nonconflicting. Two commitments with a common debtor are conflicting if the debtor cannot discharge them simultaneously. In the case of nonconflicting commitments, an agent can simultaneously be involved as a debtor in two active commitments. Conflicting commitments may be resolved by the corresponding context Orgs when they receive escalations: either from a debtor when the conflict is detected or from one of the creditors when the debtor fails to discharge the commitment.

Dependencies among commitments. Commitments formed in the context of overlap-
ping Orgs may be dependent on each other. For example, the discharge of one commitment can cause the discharge of another commitment. Such dependencies among commitments can be captured using *conditional commitments*, where the precondition of the conditional commitment may specify an operation on another commitment. Dependencies may sometimes lead to violations due to situations such as time outs. These are handled by the context Orgs listening to the escalations.

**Handling escalations in overlapping Orgs.** An overlapping Org can handle escalations in different ways. The Org can resolve it locally if possible, or it can release the common debtor and forward the escalation to an higher Org who can now take compensatory actions for the commitment, or the two overlapping Orgs may communicate with each other, which may result in an understanding to put one of the commitments on hold, until the other commitment is discharged.
Chapter 3

Organizational Design Patterns

We introduced service engagements in Chapter 1 and showed how they are characterized by autonomous parties coming together in a contractual arrangement to share resources or carry out tasks for one another. Chapter 2 provides a high-level conceptual model for service engagements, which is centered on the notion of agents. Orgs are multiagent systems, each individual and Org being modeled (recursively) as an agent. Commitments among participants, being the atoms of service engagements are created and manipulated as a service engagement progresses. Orgs apply in modeling service engagements by providing an architectural home for agreements among the participants. Specifying an engagement directly via commitments is unwieldy. Fortunately, engagements exhibit several key repeating patterns.

This chapter formalizes such patterns in terms of roles, commitments, policies, and allied concepts to provide a powerful, yet simple vocabulary to specify engagements. Each pattern reflects a distinct aspect of some component of a service engagement. We also develop well-formedness criteria on how the patterns may be combined into engagements.

Agents participating in a service engagement exhibit rich policies to govern their interactions. Agents simultaneously involved in multiple commitments may potentially conflict because of timing or resource constraints. Hence agents may seek recourse by escalating the commitments to their context Org. Such decisions are formulated in the agents’ policies. For example, a context Org receiving an escalate may apply its escalate handling policy before deciding its action. Chapter 4 provides an in-depth study of enactment of policies. The
challenge of implementing using policies is not about what the policies are, but where they are placed and the vocabulary of interactions and relationships to which they apply.

This chapter contributes to policy research an approach whereby service engagements can be specified such that the key interactions among the stakeholders are identified, and policy points are identified where each stakeholder can place its policies for carrying out desired interactions. The approach naturally supports encoding some important rules of encounter for the service engagements at hand in the nature of qualifications under which a design primitive can be applied as well as the ramifications of proceeding with an interaction in a certain manner. Further, this chapter contributes to services research, a set of general-purpose computational abstractions and primitives by which service engagements can be defined.

### 3.1 Organizational Design Patterns

Designing a service engagement so that the right interactions ensue is difficult because what is right depends on what the stakeholders want. Specifying a service engagement involves designing an Org in which the engagement will be enacted. This includes specifying the member roles and their capabilities, the relationships among the roles, the policies of each role, the constraints on the roles and their capabilities, the commitments required of the roles. Service engagements have elements that fall into repeating patterns that we can formalize and use for specifying engagements. A design pattern is a generic solution to a problem that occurs commonly [Gamma et al., 1995]. An Org (design) pattern specifies distinct, commonly occurring elements of service engagements. Org patterns simplify the specification of flexible service engagements.

A service engagement is designed by specifying two or more roles and applying one or more Org patterns on them. Syntactically, a service engagement design is just like an Org pattern, so for brevity we treat them alike. Figure 3.1 describes our Org pattern model. Since we treat an Org pattern and a service engagement alike, an Org pattern may recursively include Org patterns to allow for a service engagement to include one or more Org patterns. An Org pattern requires two or more roles, and each role has an associated capability. An Org pattern describes a set of qualifications. Each pattern includes a set of policy points for each role, indicating where its policies apply. To participate in a service
engagement, agents select and instantiate one or more roles specified in the corresponding service engagement design. The actual policies are authored when a service engagement is created. The policies are computed by agents playing the roles at runtime to decide on the actions needed to enact the service engagement. For each policy point, a pattern specifies the ramifications such as any commitments that result when the corresponding decision is made.

Table 3.1 describes the Org pattern schema and explains the components of the above conceptual model. We present the descriptions in English, but each of these can be expressed in XML or an executable language such as Jess that can be used in our prototype implementation.
Table 3.1: Org pattern specification schema

<table>
<thead>
<tr>
<th>Name:</th>
<th>A unique name for the Org pattern</th>
</tr>
</thead>
<tbody>
<tr>
<td>Roles and Capabilities:</td>
<td>The roles and their capabilities relevant to the Org pattern</td>
</tr>
<tr>
<td>Purpose and Applicability:</td>
<td>The pattern’s purpose and situations where it is applicable</td>
</tr>
<tr>
<td>Qualifications:</td>
<td>A set of constraints on the roles (their capabilities and commitments) for the pattern to be applied</td>
</tr>
<tr>
<td>Scenarios:</td>
<td>A description of how the roles interact</td>
</tr>
<tr>
<td>Policies and Ramifications:</td>
<td>A set of policy points of the roles and the ramifications for each incremental interaction</td>
</tr>
<tr>
<td>Known uses:</td>
<td>Example scenarios of the pattern usage</td>
</tr>
</tbody>
</table>

3.1.1 Example Org Patterns

The patterns below use the following notation. For agents $A$ and $B$, their context Org is written as $\overline{AB}$. $D$ and $Cr$ are the debtor and creditor roles of a commitment $C_{DCr}$, which has a discharge condition $\phi$ and a context Org role $\overline{DCr}$. $\text{Cap}_X$ is the set of capabilities of role $X$. $\phi \in \text{Cap}_X$ means that role $X$ has the required capability to achieve $\phi$. $\text{Res}_{DCr}$ is the set of resources relevant to the commitment $C_{DCr}$.

Delegation of a commitment occurs when a debtor committed to provide a service to a creditor delegates the commitment to a delegatee who is capable of performing the service. Delegation is thus governed by the policies of the delegator, the delegatee, and the creditor of the commitment being delegated. Similarly a creditor of a commitment can assign the commitment to an assignee. These scenarios are handled by the delegation and the assignment patterns. Tables 3.2 and 3.3 provide these two pattern specifications. Figures 3.2(a) and (b) describe the delegation pattern and the assignment pattern scenarios respectively. Here, the circles are the roles; dotted edges with solid arrows indicate existing commitments. The solid edges with arrows indicate an operation in the given direction. The relevant policy points of the pattern are indicated next to the respective nodes. Other constraints are shown next to the nodes. Figures 3.2 and 3.3 show the patterns as they may be used to specify an engagement. These reflect design-time relationships among various roles as in an engagement specification, and may be enacted in multiple ways.
Table 3.2: Delegation pattern specification

| Name: **Delegation** (debtor \(D\), creditor \(Cr\), delegatee \(D'\)) |
| Purpose and applicability: The debtor of a commitment delegates it to a delegatee who may accept the delegation, thus creating a new commitment with the delegatee as the new debtor |
| Qualifications: |
| \(C_{DCr} : C(D, Cr, DCr, \phi)\): the commitment to be delegated exists |
| \(C_{D'D} : C(D', D, DCr, \phi')\): \(D'\) is committed to \(D\) to accept \((\phi')\) the delegations of commitments that require \(\phi\) |
| \(Cn1: \phi \in Cap_{D'}\): the delegatee has the required capability to satisfy the commitment |
| \(Cn2: D' \neq Cr\): the delegatee is not the creditor |
| Scenarios: |
| In Figure 3.2(a), \(D\) sends a delegate request to \(D'\), \(D'\) accepts the delegate and creates a new commitment \(C_{D'Cr}\) within a newly created context \(Org\ D'Cr\) |
| Policies and Ramifications: |
| **P1** (\(D\)'s policy point): \(D\) delegates the commitment to \(D'\) |
| Ramification: \(D\) grants access to the resources \(Res_{DCr}\) to \(D'\) |
| **P2** (\(D'\)'s policy point): \(D'\) accepts the delegate from \(D\) |
| Ramification: \(C_{D'Cr} : C(D', Cr, DCr, \phi)\) is created |
| **P3** (\(Cr\)'s policy): \(Cr\) accepts the newly created commitment. Otherwise, the new commitment is released |
| Known uses: |
| A merchant delegates shipping a product to a shipper |

Figure 3.2: Delegation, assignment, and escalation patterns
Table 3.3: Assignment pattern specification

<table>
<thead>
<tr>
<th>Name: <strong>Assignment</strong> (debtor $D$, creditor $Cr$, assignee $Cr'$)</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Purpose and applicability:</strong></td>
</tr>
<tr>
<td>This creditor of a commitment may assign it to an assignee, resulting in the debtor creating a new commitment to the assignee as the creditor</td>
</tr>
<tr>
<td><strong>Qualifications:</strong></td>
</tr>
<tr>
<td>$C_{DCr} : C(D, Cr, DCr, \phi)$: the commitment that is to be assigned exists</td>
</tr>
<tr>
<td>$C_{Cr'C} : C(Cr', Cr, CrCr', \phi')$: $Cr'$ is committed to $Cr$ to accept ($\phi'$) any assigns</td>
</tr>
<tr>
<td><strong>Scenarios:</strong></td>
</tr>
<tr>
<td>In Figure 3.2(b), $Cr$ sends a assign request to $Cr'$, $Cr'$ accepts it; $D$ creates a new commitment $C_{DCr'}$ within a newly created context Org $DCr'$</td>
</tr>
<tr>
<td><strong>Policies and Ramifications:</strong></td>
</tr>
<tr>
<td>P1 ($Cr$’s policy point): $Cr$ assigns the commitment to $Cr'$</td>
</tr>
<tr>
<td><strong>Ramification:</strong> $Cr$ grants access to the resources $Res_{DCr}$ to $Cr'$</td>
</tr>
<tr>
<td>P2 ($Cr'$’s policy point): $Cr'$ accepts the assign from $Cr$</td>
</tr>
<tr>
<td><strong>Ramifications:</strong></td>
</tr>
<tr>
<td>$Cr'$ grants access to the resources $Res_{DCr'}$ to $D$</td>
</tr>
<tr>
<td>P3 ($D$’s policy point): $D$ creates a new commitment to $Cr'$</td>
</tr>
<tr>
<td><strong>Ramification:</strong> $C_{DCr'} : C(D, Cr', DCr', \phi)$ is created</td>
</tr>
<tr>
<td><strong>Known uses:</strong></td>
</tr>
<tr>
<td>A bank assigns a customer’s loan collection to a collection agency</td>
</tr>
</tbody>
</table>
Delegated commitments may potentially fail to discharge and get canceled. Under such circumstances, the original commitment may be reactivated. Other resolution options include redelegating the original commitment to a new debtor, or preempting the failing delegatee from other service engagements that may potentially be the cause for it to cancel the commitment. This is made possible by the escalation pattern. Table 3.4 provides the escalation pattern specification.

Table 3.4: Escalation pattern specification

| Name: Escalation (debtor D, creditor Cr) |
| Purpose and applicability: |
| Commitments may be canceled or otherwise violated. Under such circumstances, the creditor or the debtor of the commitment may send escalations to the context Org. |
| Qualifications: |
| $C_{DCr}: C(D, Cr, DCr, \phi)$ |
| Either of the following should occur: |
| Cn1: D cancels $C_{DCr}$ |
| Cn2: Cr releases D from $C_{DCr}$ |
| Scenarios: |
| In Figure 3.2(c), D cancels the commitment. D or Cr may send an escalate to $DCr$. $DCr$ takes an action based on its escalate handling policy, either locally resolving it, or may forward the escalate up to its parent Org in the Org hierarchy |
| Policies and Ramifications: |
| P1 (D’s policy point): If D cancels $C_{DCr}$, it escalates it to $DCr$ |
| P2 (Cr’s policy point): If Cr releases D from $C_{DCr}$, it escalates it to $DCr$ |
| P3 ($DCr$’s policy point): $DCr$ accepts escalates from D or Cr $DCr$’s escalate handling policy would determine if the escalate is forwarded up the Org hierarchy, or if $DCr$ locally handles the escalated commitment. |
| $C_M: C(Dm, Cr, DmCr, \phi)$, is the resulting commitment after resolution, where Dm is a compensating debtor |
| Known uses: |
| A customer complains if an ordered shipment does not arrive |

Service engagements often result in some agents being simultaneously involved in multiple commitments. This may lead to conflicting situations and hence there is a need for either preempting some agents, or reallocating to handle conflicts. The example discussed in Section 1.4.2 shows preemptive scheduling as a way to resolve conflicting situations. The
preemption and reallocation patterns capture these scenarios. Tables 3.5 and 3.6 describe the preemption and reallocation pattern specifications.

Collaboration among several enterprises is commonly observed resulting in a set of services being offered under a single roof. Service engagements are created with external parties by a single Org that encloses all the collaborating parties. Depending on the types of the commitments that are created due to the service engagement, the commitments are delegated to any of the collaborating parties with matching capabilities. The collaboration pattern captures this scenario. Table 3.7 provides the collaboration pattern specification.

For brevity we do not show other patterns in detail. The pattern $\text{Accept}(A, B)$ captures the scenario where $B$ accepts a message received from $A$ and performs the necessary actions based on its policies. This pattern combines with the patterns discussed above. In essence, it captures a relationship under which the accepting role gives up its autonomy with respect to the specified action. Similarly design patterns can be described for the commitment manipulation scenarios such as creation, cancellation, discharge, and release. More sophisticated patterns include Backup, Separation of Duties, and Least Privilege.
Table 3.5: Preemption pattern specification

| Name: Preemption \((D, Crl, Crh, Pr)\) Here, \(Crl\) is a lower priority creditor than \(Crh\), and \(Pr\) is the preemting role |
| Purpose and applicability: |
| To cancel a commitment based on conflicting demands |
| Qualifications: |
| \(C_{DCrl} : C(D, Crl, DCrl, \phi_1)\): the lower priority commitment of \(D\) exists |
| \(C_{DCrh} : C(D, Crh, DCrh, \phi_2)\): the higher priority commitment of \(D\) exists |
| \(Cn1: Pr\) has set a higher priority for \(C_{DCrh}\) than \(C_{DCrl}\) |
| Scenarios: |
| In Figure 3.3(a), \(D\) is unable to discharge both the commitments \(C_{DCrl}\) and \(C_{DCrh}\) simultaneously. \(D\) is preempted by \(Pr\) from the commitment \(C_{DCrl}\) and \(D\) can now discharge the commitment \(C_{DCrh}\) to the creditor \(Crh\). \(Pr\) would have either directly received an escalation from \(D\), or \(Pr\) would be a member of a context Org who received the escalation. Sometimes, \(Pr\) would be the delegator, who preempts the delegatee \(D\) from a failing commitment. |
| Policies and Ramifications: |
| \(P1\) (Pr’s policy point): \(Pr\) sends a preemption request to \(D\) to cancel \(C_{DCrl}\) |
| \(P2\) (D’s policy point): \(D\) receives a preemption request from \(Pr\) to cancel \(C_{DCrl}\) |
| Ramification: \(P1\) and \(P2\) result in \(C_{DCrl}\) being canceled by \(D\). This may in turn lead to new escalations of \(C_{DCrl}\), which will require a new debtor |
| Known Uses: |
| A physician is preempted from his consultation hours to attend to a medical emergency |
Table 3.6: Reallocation pattern specification

<table>
<thead>
<tr>
<th>Name: <strong>Reallocation</strong> ((D, Cr, Db, Dm)) Here, (Db) is the busy delegatee, and (Dm) is the compensating delegatee</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Purpose and applicability:</strong> Reallocation happens when a delegatee that cancels a delegated commitment is released and the original commitment is redelegated to another agent. Here, a cancel is implied on timeouts (when a violation occurs). This occurs when an agent is involved in multiple contracts simultaneously</td>
</tr>
<tr>
<td><strong>Qualifications:</strong></td>
</tr>
<tr>
<td>(C_{DCr} : C(D, Cr, \overline{DbCr}, \phi)): the original commitment that gets delegated exists</td>
</tr>
<tr>
<td>(C_{DbCr} : C(Db, Cr, DbCr, \phi)), the delegated commitment that fails to discharge</td>
</tr>
<tr>
<td>(Cn1): The busy delegatee (Db) fails to discharge (C_{DbCr})</td>
</tr>
<tr>
<td>(Cn2): (\phi \in Cap_{Db}), (Cn3): (\phi \in Cap_{Dm}): both the delegatees have the required capability to satisfy the commitment</td>
</tr>
<tr>
<td>(Cn4): (Db \neq Dm \neq Cr): the delegatee cannot be the creditor</td>
</tr>
<tr>
<td><strong>Scenarios:</strong></td>
</tr>
<tr>
<td>In Figure 3.3(b), (D) releases (Db) from the delegated commitment ((C_{DbCr})), and it redelegates the original commitment to the compensating delegatee (Dm). Here, we assume that the creditor accepts the new delegated commitment from (Dm). Reallocation may be initiated by (D) when requested by (DCr) that might have received escalations forwarded by the context Org (DbCr)</td>
</tr>
<tr>
<td><strong>Policies and Ramifications:</strong></td>
</tr>
<tr>
<td>(P1) ((D)'s policy point): When (D) knows about (Cn1), it releases (Db) from (C_{DbCr}) and delegates (C_{DCr}) to (Dm)</td>
</tr>
<tr>
<td>(P2) ((Db)'s policy point): When (Db) receives a release message from (D), it accepts it and (C_{DbCr}) is released</td>
</tr>
<tr>
<td>(P3) ((Dm)'s policy point): (Dm) accepts the delegate from (D)</td>
</tr>
<tr>
<td><strong>Ramification:</strong> Upon reallocation, the following commitment applies:</td>
</tr>
<tr>
<td>(C_{DmCr} : C(Dm, Cr, \overline{DmCr}, \phi)), the new delegated commitment</td>
</tr>
<tr>
<td><strong>Known Uses:</strong> In the hospital scenario, when a doctor scheduled to attend to patient (A) in a hospital needs to attend to another (critical) patient (B), the hospital may reallocate another doctor in his place to see patient (A)</td>
</tr>
</tbody>
</table>
Table 3.7: Collaboration pattern specification

| Name: Collaboration (collaborating roles \{R_1, \ldots, R_n\}, collaboration Org \(O_C\), creditor \(Cr\)) |
| Purpose and applicability: Two or more agents may collaborate to provide a set of services as a team. The pattern is applicable in these scenarios, where the collaboration is handled by an Org that is responsible for creating any contracts with other agents. The Org delegates the commitments created to the collaborating agents based on the delegation policies |
| Qualifications: C1: \(C_{\rightarrow} : C(O_C, Cr, O_Cr, \phi_{\rightarrow})\): A commitment from \(O_C\) to provide a service to \(Cr\) exists C2: \(C_{<\rightarrow} : C(Cr, O_C, O_Cr, \phi_{<\rightarrow})\): A commitment from \(Cr\) to \(O_C\) exists to provide timely payments for the service received Commitments \(C_1, \ldots, C_n\) exist between the collaborating roles and \(O_C\) to enable the delegations and assignments required for handling the commitments created with external creditors Cn1: \(\phi_{\rightarrow} \in Cap_{R_i}\) |
| Scenarios: The commitment \(C_{\rightarrow}\) is delegated by \(O_C\) to one of the collaborating agents (say, \(R_i\)) depending on the delegation policy of \(O_C\). Similarly, the commitment \(C_{<\rightarrow}\) is assigned by \(O_C\) to \(R_i\) |
| Policies and Ramifications: P1 (\(O_C\)’s policy point): \(O_C\) has a delegation handling policy that enables the delegation of the commitment \(C_{\rightarrow}\) to \(R_i\) based on matching \(\phi_{\rightarrow}\) to \(R_i\)’s capabilities. Similarly \(O_C\) applies a policy to assign \(C_{<\rightarrow}\) to \(R_i\) P2 (\(R_i\)’s policy point): \(R_i\) has policies to accept any delegation or assign of commitments sent by \(O_C\) Ramification: Upon delegation, the following commitment applies: \(C_{R_iCr} : C(R_i, Cr, O_Cr, \phi_{\rightarrow})\) Upon assignment, the following commitment applies: \(C_{CrR_i} : C(Cr, R_i, O_Cr, \phi_{<\rightarrow})\) |
| Known uses: In virtual enterprises two or more enterprises get together to handle service contracts with customers. These organizations share resources, divide their responsibilities, and work together forming a collaboration |
3.1.2 An Example Service Engagement

A service engagement is specified using the above patterns. Each engagement specification follows the template of Table 3.1. Here, the qualifications are assertions corresponding to the patterns applied on appropriate roles. All policy points and ramifications are inherited from the patterns. A service engagement specification not only states the patterns but also specifies a set of key services captured as commitments and other constraints. A service engagement is instantiated when agents instantiate the relevant roles provided all qualifications are met. The agents supply policies for each policy point and are subject to the stated ramifications.

We consider a simplified version of the service provisioning example scenario presented in Section 1.4.2 and formalize the NOAA–NL service engagement using the above approach. DOC and DOE are not considered here. Figure 3.4 illustrates this simplified scenario. NOAA contains the National Hurricane Center (NHC). Likewise, NL contains Argonne (ANL) and Oak Ridge (ORNL). Service agreements among organizations can be delegated, assigned, or otherwise manipulated to result in service agreements among their members. Org-NOAA-NL is the Org for the service engagement between NOAA and NL.

Roles and Role Instantiations: Service-Providers-Org (SPO), Service-Provider (SP), Service-Consumer-Org (SCO), Service-Consumer (SC). These roles are instantiated by the agents as follows: National Labs (SPO), ANL (SP), NOAA (SCO), NHC (SC).

Key Services as Commitments: Commitment C₃: C(SP, SC, Org-SP-SC, φ), where φ means the compute service is provisioned successfully. Commitment C₁ is first created that captures the engagement between SPO and SCO within Org Org-SPO-SCO. SCO assigns C₁
to SC to create an assigned commitment $C_2$. $C_2$ is now delegated by SPO to SP to create $C_3$.

![Figure 3.5: Service engagement instantiations](image)

**Qualifications:** These include the delegation, assignment, escalation, and the preemption patterns as illustrated in Figure 3.5 and described below:

**Assignment (SPO, SCO, SC):** SCO assigns $C_1$ to SC. After the assign, SPO makes a new commitment $C_2$ to SC within Org-SPO-SC.

**Delegation (SPO, SC, SP):** Commitment $C_2$ from SPO to SC is delegated to SP. A new commitment $C_3$ is created with SP as the new debtor.

**Escalation (SP, SC, Org-SP-SC):** Commitment $C_3$ from SP to SC exists within Org-SP-SC. SP or SC can send escalates to the context, Org-SP-SC.

**Preemption (SP, SCL, SC, SPO):** Here, SCL is the service consumer, who is the creditor of the lower priority commitment $C_4$, and SC is the creditor of the higher-priority commitment $C_3$. SPO is the preempts Org that preempts SP from the lower priority commitment.

### 3.2 Formal Results

The roles in an Org are linked in different ways depending on the set of Org patterns instantiated by the Org. We describe basic and composite Org patterns and provide a
classification of the different ways of linking the roles of an Org. We arrive at some well-formedness criteria on how the patterns may be combined in service engagements.

![Org structure with role links](image)

Figure 3.6: Org structure with role links

### 3.2.1 Structure Based on Design Patterns

A basic Org pattern involves roles and manipulations of a single commitment and its delegated or assigned forms. Delegation, assignment, escalation, creation, cancellation, release, and discharge are examples of basic Org patterns. The Org patterns demonstrate how the various roles are linked.

**Definition 9** A role link \(\langle P \rightarrow_X Q \rangle\) means that roles \(P\) and \(Q\) are related for a purpose specified by type \(X\).

The above patterns yield the following role link types.

- **Delegation link** \(\langle D \rightarrow_{De} D' \rangle\): A delegator \(D\) can delegate a commitment to \(D'\), a delegatee.

- **Assignment link** \(\langle Cr \rightarrow_{A} Cr' \rangle\): A creditor \(Cr\) can assign a commitment to an assignee \(Cr'\).
Table 3.8: Representative pattern roles and role links

<table>
<thead>
<tr>
<th>Pattern</th>
<th>Roles</th>
<th>Role links</th>
</tr>
</thead>
<tbody>
<tr>
<td>Delegation</td>
<td>$D, Cr, D'$</td>
<td>$\langle D \rightarrow_C Cr \rangle$, $\langle D \rightarrow_{De} D' \rangle$, $\langle D' \rightarrow_C Cr \rangle$</td>
</tr>
<tr>
<td>Assignment</td>
<td>$D, Cr, Cr'$</td>
<td>$\langle D \rightarrow_C Cr \rangle$, $\langle Cr \rightarrow_{A} Cr' \rangle$, $\langle D \rightarrow_C Cr' \rangle$</td>
</tr>
<tr>
<td>Escalation</td>
<td>$D, Cr, DCr$</td>
<td>$\langle D \rightarrow_C Cr \rangle$, $\langle D \rightarrow_E DCr \rangle$, $\langle Cr \rightarrow_E DCr \rangle$</td>
</tr>
<tr>
<td>Creation, Cancellation, Discharge</td>
<td>$D, Cr$</td>
<td>$\langle D \rightarrow_C Cr \rangle$,</td>
</tr>
<tr>
<td>Release</td>
<td>$D, Cr, Rr$</td>
<td>$\langle D \rightarrow_C Cr \rangle$, $\langle Rr \rightarrow_R D \rangle$</td>
</tr>
</tbody>
</table>

- **Commitment link** $\langle D \rightarrow_C Cr \rangle$: $D$ is committed to $Cr$. This link enables $D$ to create, cancel, and discharge the commitment.

- **Escalation link** $\langle R \rightarrow_E DCr \rangle$: A role $R$, which could be either $D$ or $Cr$, can escalate $C_{DCr}$ to the context $DCr$.

- **Org Escalation link** $\langle Oc \rightarrow_O Op \rangle$: Org $Oc$ is a child of $Op$, and can forward an escalate to $Op$.

- **Release link** $\langle Rr \rightarrow_R D \rangle$: A role $Rr$ (creditor or context) can release a debtor $D$ from its commitment.

Table 3.8 shows the roles and role links introduced by Org patterns. Figure 3.6 (the role DM is not shown) describes the Org structure and the role links resulting from the enactment of the service engagement described in Section 3.1.2.

**Composition of Organizational Patterns** The basic Org patterns combine to form composite patterns. Preemption and Reallocation are compositions of two or more basic patterns. Below, $\oplus$ denotes pattern composition.

**Observation 1** Preemption composes the release and two copies of the create pattern. It involves two or more commitment links from the same debtor role ($D$) to two different creditors, $Crl$ and $Chl$. There exists a role ($Pr$) that preempts (releases) the debtor ($D$) from the commitment to $Crl$. 
\begin{equation}
\text{Preemption}(D, Crl, Crh, Pr) = \text{Creation}(D, Crl) \oplus \text{Creation}(D, Crh) \oplus \text{Creation}(D, Crl)
\end{equation}

\begin{equation}
\text{Release}(D, Crl, Pr)
\end{equation}

\begin{equation}
\text{Role-links} = \{\langle D \rightarrow C Crl \rangle, \langle D \rightarrow C Chl \rangle, \langle Pr \rightarrow R D \rangle\}
\end{equation}

Observation 2  Reallocation composes the release and two copies of the delegation pattern. It involves delegatee roles \(Db\) and \(Dm\), debtor \(D\), and creditor \(Cr\).

\begin{equation}
\text{Reallocation} = \text{Release}(Db, Cr, D) \oplus \text{Delegation}(D, Cr, Db) \oplus \text{Delegation}(D, Cr, Dm)
\end{equation}

\begin{equation}
\text{Role-links} = \{\langle D \rightarrow C Cr \rangle, \langle D \rightarrow De Db \rangle, \langle Db \rightarrow C Cr \rangle, \langle D \rightarrow De Dm \rangle, \langle Dm \rightarrow C Cr \rangle, \langle D \rightarrow R Db \rangle\}
\end{equation}

3.2.2 Sound Service Engagements

The specification of a service engagement that composes several Org patterns might potentially result in erroneous situations at runtime. We describe some problem scenarios and provide guidelines for specifying a sound service engagement specification.

Postulates on Escalation.  Delegated and assigned commitments may be violated. Hence a sound service engagement specification should include ways for the creditor and the debtor of new commitments to escalate any problem to their context Orgs. Any commitment should always provide for escalation to its own context Org. The following postulates capture these requirements as guidelines. Here, \(R_1, R_2, R_3, R_4\) are roles in a service engagement and Org-\(R_1-R_2,\) Org-\(R_3-R_2,\) and Org-\(R_3-R_4\) are roles for the context Orgs of the commitments created in this service engagement.

Postulate 1  Delegation and assignment pattern instantiations should be accompanied by escalation pattern instantiations in a sound service engagement.

In the case of delegation pattern, if \(\langle R_1 \rightarrow C R_2 \rangle, \langle R_1 \rightarrow De R_3 \rangle,\) and \(\langle R_3 \rightarrow C R_2 \rangle\) exist, then a sound service engagement should have \(\langle R_3 \rightarrow E \text{Org-}R_3-R_2 \rangle, \langle R_2 \rightarrow E \text{Org-}R_3-R_2 \rangle, \langle \text{Org-}R_3-R_2 \rightarrow O \text{Org-}R_1-R_2 \rangle\). In Figure 3.6, Delegation(NL, NHC, ANL) is accompanied
by \textit{Escalation}(ANL, NHC, Org-NHC-ANL) and \textit{Assignment}(NL, NOAA, NHC) is accompanied by \textit{Escalation}(NL, NHC, Org-NHC-NL). If these were missing, a delegated or assigned commitment may be violated with no recourse leading to failure of the overall engagement.

**Postulate 2** If $⟨R_1 \rightarrow_C R_2⟩$ exists, then a sound service engagement should have $⟨R_1 \rightarrow_E \text{Org}\,-\,R_1\,-\,R_2⟩$ and $⟨R_2 \rightarrow_E \text{Org}\,-\,R_1\,-\,R_2⟩$, i.e., a commitment creation should be accompanied by an escalate pattern instantiation.

In Figure 3.6, \textit{Creation}(NL, NOAA) is accompanied by \textit{Escalation}(NL, NOAA, Org-NOAA-NL). Otherwise, the context, Org-NOAA-NL which scopes the commitment between NL and NOAA may not be able to take compensatory actions for a failing commitment.

**Postulates on Delegation and Assignment.** The delegation and assignment patterns are irreflexive and asymmetric. This is an important guideline to consider while instantiating the delegation and the assignment patterns to avoid nonterminating service engagements, and hence required for \textit{completeness} of a service engagement.

**Postulate 3** If $⟨R_1 \rightarrow_{+} De R_3⟩$ exists for a commitment, then a \textit{complete} service engagement should not have $⟨R_3 \rightarrow_{+} De R_1⟩$ for the same commitment.

Here, we consider chains of delegations. $⟨R_1 \rightarrow_{+} De R_3⟩$ implies that $R_1$ may delegate to $R_3$ through a chain of one or more steps. For example, in Figure 3.6 $⟨\text{NL} \rightarrow_{De} \text{ANL}⟩$ and $⟨\text{ANL} \rightarrow_{De} \text{NL}⟩$ cannot both exist. If both exist, then it may lead to nonterminating service engagements because of cyclic delegations. Similarly $⟨R_1 \rightarrow_{A} + R_3⟩$ and $⟨R_3 \rightarrow_{A} + R_1⟩$ cannot both exist.

In a service engagement specification, if multiple delegations from different delegators of the same Org happen to a common delegatee, the delegated commitments may conflict. Hence such delegations are not desirable. But such situations are unavoidable in practice, because some roles are often involved in multiple commitments simultaneously. To ensure soundness in such Orgs and to handle conflicting situations, we need to allow for the preemption (release) of the common delegatee from lower priority commitments.
Postulate 4 \( \langle D_1 \rightarrow_{De}^+ D' \rangle \) and \( \langle D_2 \rightarrow_{De}^+ D' \rangle \) cannot simultaneously exist in a sound service engagement, where \( D_1 \) and \( D_2 \) are distinct roles in the same Org, and \( D' \) is the common delegatee, unless, it is also accompanied by the release link \( \langle Pr \rightarrow_{R}^+ D' \rangle \) (indicating a chain of releases in one or more steps), where \( Pr \) is the preemining role. Here \( Pr \) is one of the delegators \( D_1 \) or \( D_2 \), or any other delegator that appears in the chain of delegations.

In the case of a one step delegation, the pattern instantiations \( \text{Delegation}(D_1, Cr_L, D') \) and \( \text{Delegation}(D_2, Cr_H, D') \) cannot both happen unless accompanied by a preemption pattern instantiation: \( \text{Preemption}(D', Cr_L, Cr_H, Pr) \). For example, \( \langle NL \rightarrow_{De}^+ ANL \rangle \) and \( \langle ORNL \rightarrow_{De}^+ ANL \rangle \) are the two delegates to ANL. These should be accompanied by a \( \langle NL \rightarrow_{R} ANL \rangle \), where NL can preempt ANL, whenever necessary. Otherwise, if the two commitments of ANL conflict, ANL may not be able to discharge the higher priority commitment, unless preempted from the other commitment.

3.3 Discussion

The services sector has long been dominant in developed economies. As services technologies and business models spread in IT, an increasing number of IT applications are taking on a services flavor. Witness the expansion of utility or autonomic computing and software as a service. Services have a natural match with policy techniques. Besides the usual challenges of policy languages and engines, services also throw up the important challenge of setting up interactions among multiple stakeholders so they can carry out a service engagement in a manner that is easy to design and configure, and yet respects their autonomy.

With this motivation, we developed a set of design patterns for specifying service engagements. This set is far from complete. Nor do we believe that a small complete set of patterns exists. A traditional programming language can be Turing complete with only a few constructs, yet books have been filled with patterns of programming. In the same vein, because service engagements exhibit enormous diversity, we expect that more and more design patterns for engagements will be identified.
Chapter 4

Policy-Based Governance in Orgs

Our multiagent conceptual model for Orgs supports a policy-based governance architecture. We described how a service engagement brings together different agents in a contractual arrangement. Because of the autonomy of the participants, governance is a better metaphor for Orgs than traditional management. Governance involves the processes by which autonomous entities agree to collaborate and share resources. We seek to improve the modeling and governance of complex Orgs through innovations in policy architecture. Section 4.1 briefly describes the concepts on which our proposed approach builds. Chapter 5 describes other literature.

4.1 Background: Traditional Policy Architectures

We describe traditional policy architectures that are generally reactive acting on a per-request basis. Our proposed approach builds on these and is inspired by them on certain key aspects. What our approach adds is an emphasis on proactive monitoring and compliance.

4.1.1 IETF Policy Framework

Figure 4.1 illustrates the IETF policy framework, which was designed for managing network resources [IETF WG, 2001; Yavatkar et al., 2000], but has a structure that applies more generally. This architecture assumes a network element (routers, hubs, or switches) where
resource allocation decisions are made. A *policy enforcement point* or PEP resides within the network element. A *policy decision point* or PDP may be within or outside the element; policy decisions are made here. A PDP may use further mechanisms or components to decide on an action.

![Network element
PEP
Policy server
PDP
Contact external policy databases, authentication servers, etc.]

Figure 4.1: The basic IETF policy framework

A typical policy transaction normally begins at a network element in the PEP with a user requesting access to network resource. The PEP consults the PDP, which retrieves applicable policies from its repositories. The PEP then applies the appropriate policy to the request at hand. The concepts of PDP and PEP are valuable in a variety of settings, including XACML (see below) as well as the proposed approach.

### 4.1.2 XACML Policy Framework

XACML, the extensible access control markup language [OASIS, 2005], extends the above framework. XACML has an access decision language used to represent a runtime request for a resource. First, a policy associated with a resource is located. Next, the attributes of the request are compared with the rules, ultimately yielding a permit or deny decision. XACML supports a runtime per-request access control mechanism and is illustrated in Figure 4.2.

When a request is received, the recipient’s policy enforcement point (PEP), forwards the request to a context handler along with some requester attributes. The context handler notifies the policy decision point (PDP). The PDP collects the required attributes from the policy information point (PIP) via the context handler. The PDP collects the appropriate policies from the policy store, and arrives at the authorization decision using requester attributes and the policy condition predicates. If multiple policies apply, the PDP combines them appropriately. The PDP finally returns the authorization decision back to
the PEP, which is conveyed to client. XACML supports obligations and the PEP ensures that these obligations are performed.

4.1.3 Grid Policies Architectures

Grid research has considered Orgs in the form of VOs (virtual organizations) and policy architectures. However, Grid policies focus on resource usage, access control, membership, and resource management. Dumitrescu et al. propose an architecture for scientific data grids that is based on usage policies [Dumitrescu et al., 2005]. The usage policy enforcement happens both at the VO level (for grid-wide policies), and at the site level (for local policies). Accordingly, this architecture includes two kinds of PEPs: S-PEP (site-level) and V-PEP (VO-level). An S-PEP ensures that only jobs satisfying local policies run at a site; others are preempted. A V-PEP schedules jobs on different sites. V-PEPs interact with S-PEPs and schedulers to enforce VO-level policies. Each PEP is supported by a monitoring distribution point (MDP), which gathers information about resource usage and policy restrictions. MDPs are distributed but can interact with other MDPs.
4.2 Org Policy Architecture Requirements

Before we describe the agent policy architecture, we study the requirements of an Org policy architecture. In a dynamic Org, the object of a policy need not be a passive entity but could also be an agent—usually one to whom the Org might have delegated some responsibility and granted some authority and visibility for the purposes of a specified family of business interactions. A desirable policy architecture would be flexible enough to capture dynamic relationships between entities in Orgs, and the context in which the entities exist. Policies should be triggered automatically leading to the creation of new Orgs, or membership in existing Orgs.

4.3 Facets of a Policy

We postulate the following facets of a policy, which are necessary for the practical policy management in an Org setting.

Creation. Policy scoping and creation corresponds to forming new Orgs, and creating contracts between existing agents within an Org. This may involve deciding agent membership criteria in an Org and selection of the appropriate Org templates while forming new Orgs.

Enactment. Policy enactment enables compliant interactions across and within Orgs. The agent actions are controlled by the policies. For example, it facilitates delegation management within the nested structures of Orgs, management of release or cancellations of commitments within an Org, enforcement of policies at each level of organization within an Org, and so on.

Monitoring. Policy monitoring can involve certain book-keeping tasks that include logging of events and reporting these events to the Org (in the case for member agents), or the members (in the case for an Org). For example, these events may include interactions with different parties and their outcomes, any event that happens in the enactment of a contract or a commitment within an Org. In practical Org settings such as a production supply chain scenario, monitoring involves reporting the events such as production delays, quantity, demand and so on.
Compliance. Policy compliance checking means verifying at runtime whether a policy has been violated. It may include the anticipation of looming problems that would cause a policy to be violated, and thus may motivate evasive action. Compliance checking is aided by the monitoring of events so that policy decisions can be taken to avoid any potential problems or violations within an Org. For the example of a production grid, compliance checking involves checking if the production policies are satisfied and taking some evasive actions such as replacing a particular production unit if its production rate and quality are poor.

The proposed approach goes beyond XACML in introducing additional architectural components geared toward monitoring and compliance. It borrows, from traditional grid policy architectures, the notions of monitoring and aggregating distributed events, and hierarchical enforcement of policies. However, we extend these notions in key respects, including:

1. event monitoring and gathering to support proactive mechanisms,
2. compliance checking aided by monitoring to enable proactive mechanisms,
3. policy modeling and handling in recursively formulated Orgs, and
4. an architecture supporting autonomy and policy compliance.

4.4 Agent Policy Architecture

Figure 4.3 describes our agent architecture. An agent contains a PEP and a PDP as in traditional policy architectures. But the PEP and the PDP provide additional features with the aid of two new components: Policy Monitoring Point (PMP), and Policy Organizational Point (POP). These are the main modules that constitute our “M-O-D-E” (PMP: M, POP: O, PDP: D, PEP: E) architecture. An agent connects to an event bus [Chappell, 2004] to send and receive events. The PMP helps to make this agent policy architecture event-based, yet proactive. Whereas a traditional PEP would merely allow or disallow a requested event based on the PDP, here, the PEP can perform an action different from any event that might have been requested. In particular, a domain action may be taken even if there is no
explicit external event specifically triggering that action; the triggering condition might be internal to the agent, such as based on its prior commitments, events stored in the history, or merely the passage of time.

**POP.** The POP manages “organizational” relationships, which are reflected mainly in the configuration store and the policies. Thus the POP potentially changes the behavior of the PDP by modifying the policies that the PDP applies. In essence, the POP “reconfigures” an Org by performing operations on the commitments that apply to the parties involved, or requesting such operations from other agents, such as a higher-level Org.

**PMP.** The PMP is crucial in enabling proactive event-driven behavior. It observes or monitors an event stream that can include *communicative actions* (described in Section 4.5.3) and environmental events. PMP captures specified event patterns from the stream(s), as instructed by the POP. Some communicative actions are *organizational* events that add or manipulate the contents of the various stores of the agent. Other events are *domain* events that include requests and responses. The PMP dispatches the organizational events to the POP, which may apply its policies and take the appropriate actions on the data stores. The
PMP sends events to the PEP for further processing. All events are stored in the history.

The PEP’s actions are relayed back to the event bus, and thus visible to its own PMP. If an action performed by the PEP involves any changes in its stores (e.g., events such as the creation or manipulation of a commitment), the PMP sends the action as an organizational event to the POP, which may perform an appropriate action on the corresponding store.

4.5 Dynamics of Service Engagements arising in Orgs

In Chapter 2, we presented an agent-based conceptual model for service engagements that are modeled as arising within Orgs. This section provides the dynamic aspects of our conceptual model for Orgs that include the descriptions of the life cycles of commitments and Orgs. This model supports our policy-based governance architecture. To accommodate the policies, each agent (an Org or an individual agent) maintains a set of policies and configuration facts. Configuration facts describe the structures in which an agent participates. They include information about their existing relationships such as commitments and contracts, organizational structure, power relationships, and so on. To support hierarchical policy enforcement, an Org is defined as follows:

**Definition 10** An Org $O$ is an agent that encloses a set of member agents $M = \{A_1, \ldots, A_n\}$, a set of policies $P$, and a set of configuration facts $C$.

Each agent monitors and records events, which correspond to environmental observations or the actions of this and other agents. Based on these events and their policies, agents modify the configuration and choose their actions. Each agent (individual or Org) that has nonnull policies instantiates a policy engine that controls its communicative and other actions, including those in response to environmental events and the communicative actions of others.

4.5.1 Creation of Orgs within Institutions

We demonstrate how Orgs are created within institutions and illustrate with an example scenario. Figure 4.4 illustrates a possible scenario. An institution $T$ already exists that
includes a repository of one or more Org templates, and a repository for zero or more instantiated Orgs. Here, agents $A_1$–$A_6$ attempt to enter the institution $T$ (by registering with $T$). $A_1$–$A_5$ qualify as members, while $A_6$ fails to get registered in $T$ (Steps 1 and 2). $A_1$ and $A_2$ wish to contract with each other, and choose an existing Org template ($O^i$), instantiate the roles defined in it, and create an instantiated Org: $O^i_1$. Similarly, $A_4$ and $A_5$ create $O^i_2$. If $O^i_1$ is open to new members, $A_3$ may enter it, provided $A_3$ is already a member of $T$, and meets the entry policies of $O^i_1$ (Step 3). $A_3$ becomes a member of $O^i_1$ due to an operation such as a delegation of a commitment by $A_1$ or $A_2$, and is situated below the delegator agent in the organizational hierarchy (not shown in Figure 4.4). Org $O^i_2$ is dissolved when the contract is either enacted successfully, or canceled (Step 4).
In simple terms, this means agents must first find their way into an institution. Once there they have established the credentials needed to enter into contracts with their peers. The institution thus provides critical functions (as needed) such as helping agents find one another, credentialing, and nonrepudiation of contracts.

### 4.5.2 Hierarchical Policy Enforcement within an Org

The hierarchical structure of an Org, coupled with the distributed nature of its members, has interesting consequences on policy enforcement. Specifically relevant policies of an Org can be pushed to its members, ensuring that a member not only complies with its local policies, but also with the policies of its ancestor Orgs. The policy architecture also supports events being propagated between an Org and its members.

Figure 4.5 represents a segment of a hierarchy with an Org and two of its children (To improve legibility the event bus is shown as a separate module (but connected) for the top Org and its two children). The event flows indicate that agents can send events to their parents, siblings, or children.

![Hierarchical enforcement schematically](image)

**Policy Types.** We define three kinds of policies: *local* \((P_L)\), *hierarchical* \((P_H)\), and
forwarded \((P_F)\). The local policies of agent \(A\) \((P_L(A))\) are policies that are set autonomously by the agent. The forwarded policies of agent \(A\) \((P_F(A))\) are the agent’s policies that it propagates to its descendants. The hierarchical policies of agent \(A\) \((P_H(A))\) are the forwarded policies derived from \(A\)’s ancestors.

For example, in our DOE–DOC scenario, DOE may have a local policy to offer the “best” graded QoS when dealing with DOC, or with DOC’s descendants. DOC may have a local policy to contact only DOE, or DOE’s descendants for any high-performance requirements. The local policies of DOE and DOC mentioned above are also their corresponding forwarded policies, and hence are propagated down to their respective descendants, including ANL and NHC. The hierarchical policies of NHC and ANL include the forwarded policies of their ancestors NOAA and DOC (for NHC), and National Labs and DOE (for ANL).

A parent Org propagates its forwarded policies to its children via communicative actions. Here, we note that \(P_F(A) \subseteq P_L(A) \cup P_H(A)\). The PMP of a child receiving the policies may trigger an organizational event that is sent to the POP. The child’s POP would (if the source is confirmed as a genuine ancestor) add the new policies to its policy store. The hierarchical policies of an agent override its local policies, in order to allow the higher-level organizations to enforce their policies on their descendants.

**Definition 11** A policy \(p_i\) overrides a policy \(p_j\) \((p_i \succ p_j)\), if an agent considers \(p_i\) over \(p_j\) to make a policy decision.

**Property 1** \(\forall p_H \in P_H(A), p_L \in P_L(A), p_H \succ p_L:\) The hierarchical policies \(p_H \in P_H(A)\) of agent \(A\) override its local policies \(p_L \in P_L(A)\).

In the DOE–DOC scenario, consider a situation when ANL has a local policy that allows offering only lower grade QoS during a particular time period. However, one of the DOE policies overrides this local policy ensuring the “best” QoS being offered to NHC at all times.
4.5.3 Policy-Based Control of Agent Actions

Enacting cross-organizational service engagements involves agents performing several actions to manage their commitments and their configuration. The policies of an agent control its actions by specifying rules that force the agent to take certain actions or respond to the actions of other agents. We consider two kinds of actions.

**Domain Actions** are certain “functional” or application operations performed by the agent. Examples include running a simulation, allocating a light path in an optical network, a grid service provisioning, and so on. Domain actions may enable the manipulations of the configuration facts. For example, discharging a commitment may involve performing one or more domain actions.

**Communicative Actions.** In an organizational setting, commitments formed among agents may be manipulated. A communicative action may specify administrative operations on such configuration facts. The execution of a communicative action might result in one or more agent domain actions. For example, the creation of a commitment may result in multiple domain actions to discharge the commitment.

**Definition 12** A communicative action is expressed as \( \text{operationName}(A, B, \Theta) \), where \( A \) is its sender, \( B \) its recipient, and \( \Theta \) its subject.

The relevant operations are domain actions, commitment operations, and administrative operations such as request, accept, deny and so on. For the commitment operations, \( \Theta \) would be the commitment on which the operation performed, or may be the policy being propagated. For example, a commitment \( C_1 \) can be created by \( A_1 \) and communicated to \( A_2 \) using the communicative action of \( \text{create}(A_1, A_2, C_1) \).

We earlier saw the different possible operations on commitments and contracts, and they can be represented by the language \( L \) described above. In addition to these two kinds of operations, there are certain administrative operations that are discussed below.

**Administrative Operations**

Administrative operations control the manipulations of commitments and contracts and hence may have the commitment operations as their content. The relevant operations suitable in our framework are listed below:
• **Request.** A communicative action sent by an agent to request an operation. For example, an agent may request another agent to create a commitment. Here, a separate policy for the create operation may apply before an agent makes a decision about the request received.

• **Accept.** An agent may accept an operation sent as a communicative action by another agent. For example, an agent can accept a request for a create operation of a commitment sent by another agent.

• **Deny.** An agent may deny an operation sent by another agent. For example, an agent can deny a request to create a commitment.

• **Monitor.** A monitor operation can happen in both directions – from an Org to its member agent, or from a member agent to its Org. It is similar to an escalate message, where certain events are exchanged within the Org. However, escalate is only from a member agent to its parent Org. But a monitor operation can be exchanged periodically by the Org or its member agents to convey any bookkeeping events. These events are helpful in our proactive policy architecture.

**Monitoring and Escalation**

The policy architecture in our framework emphasizes the proactive aspects of policy management. Monitoring and compliance checking are the two key features of our framework that enable proactive policy-based governance. Monitoring of events is essential to support proactive policy compliance checking. The two administrative operations defined for an agent in our framework namely, monitor and escalate handle the necessary bookkeeping tasks. A monitor or an escalate can be sent by the PEP or POP, depending on the events triggered by the PMP.

PMP of an agent may also store some events received from the communicative actions sent by other agents. For example, when an Org agent receives an escalation regarding the violation or cancellation of a commitment, the Org’s POP can enforce an action by releasing the debtor of the commitment, and can assign another debtor if required.

An example scenario of a proactive mechanism as demonstrated by an Org is observed in a supply chain scenario. An Org formed between multiple supply chain components receives an escalation from one of the components indicating a problem in production.
The Org can send a monitor communicative action indicating this problem to the other production units in the supply chain that depend on this faulty production unit. These other units can act in a proactive manner and take some actions such as reduced production or stop their production to avoid losses.

The enactment of a service engagement is carried out by manipulating the associated commitments. We present below the dynamics of the enactment of service engagements by describing the life cycles of commitments and their context Orgs arising within an institution.

4.5.4 Enactment of Commitments

We describe how the commitments are enacted by describing their life cycle. We extend and simplify the life cycle for commitments proposed by Verdicchio and Colombetti [Verdicchio and Colombetti, 2002] and provide four states for commitments: *active* ($A$), *done* ($D$), *violated* ($V$), and *pending* ($P$). However, our life cycle model includes transitions such as *delegate*, *assign*, and *escalate* across commitments which enables us to address organizational considerations more naturally. The following describe the commitment operations.

**Create.** A newly created commitment is *active*.

**Discharge.** An *active* commitment changes to *done* when its condition is brought about.

**Release.** A commitment is *done* when it is released by the creditor or the context Org.

**Cancel.** A commitment is *violated* when it is canceled.

**Assign and Delegate.** A commitment being delegated by the debtor (assigned by the creditor) results in a new commitment in the *active* state with a new debtor (creditor) and the same creditor (debtor). The original commitment becomes *pending*.

**Escalate.** A delegated or assigned commitment that is *violated* results in an escalate. An escalate moves the original commitment from *pending* back to *active*.

Figure 4.6 shows the commitments operations and the corresponding state transitions. Here, when a commitment $C_i$ in *active* state is delegated or assigned, $C_{i+1}$ is the new commitment created, and similarly the delegate or assign of $C_{i-1}$ results in the creation of $C_i$. $D$, $Cr$, and $O$ are the debtor, creditor, and context of $C_i$ respectively. The nodes indicate the states of commitments. Labels of the edges indicate the commitment operations.
Figure 4.6: The life cycle of commitments

For example, the label $D$: cancel indicates that the debtor $D$ canceled the commitment $C_i$ resulting in it becoming violated.

**Chain of commitments**

A *chain of commitments* is created due to a series of delegate and assign operations on a commitment, resulting in the formation of new commitments, each with a new debtor or a new creditor. The discharge condition of the commitment remains the same. However, the new commitment can be formed within the context of a new dynamic Org that is created within the context Org of the original commitment. This simplifies computation with the new Org scoping the new commitment. This chain completes when the last commitment becomes done or violated.

**Definition 13** A *chain of commitments* is a nonempty, finite sequence of commitments $H_n = \langle C_1, \ldots, C_i, \ldots, C_n \rangle$, where $C_1$ is the original commitment and $C_{i+1}$ is formed by a delegate or assign operation on $C_i$.

A new commitment is appended to the chain every time the final active commitment is assigned or delegated. The new final commitment in the chain is the current active
commitment, while the preceding commitments are pending.

**Definition 14** A chain of commitments $H_n$ is

- **active**, when some $C_i \in H_n$ is active;
- **complete**, when some $C_i \in H_n$ is done; and
- **violated**, when $C_1 \in H_n$ is violated.

For the completion of the enactment of a service contract that includes the commitment $C_1$ from $H_n$, it is essential for the chain $H_n$ to be finite and complete.

Figure 4.7 illustrates the enactment of a commitment in the example scenario of Figure 2.6. $C_S$, from ABC to Alice, is formed within the context Org $O_1$. ABC delegates $C_S$ to DHL, which results in the creation of a commitment $C_2$ with DHL as the new debtor, and $O_2$ as the new context Org, but with the same creditor Alice. The new Org $O_2$ is formed within $O_1$ to enable the correct treatment of escalations and releases. Similarly
$O_3$ is the new Org created within $O_1$, when DHL delegate $C_2$ to DHL-USA resulting in a commitment $C_3$. Here, please note that all the context Orgs $O_i$ are generated at runtime. Here the resulting chain of commitments is $H_3 = \langle C_1, C_2, C_3 \rangle$.

The following describe the state transitions of a commitment that occur due to the operations occurring in a chain of commitments.

**Release ($C_{i-1}$)**. When a commitment ($C_{i-1}$) is released, any delegated or assigned commitment ($C_i$) next in the chain of commitments is also released. Hence $C_i$ in *active* or *pending* state changes to *done*.

**Discharge ($C_{i+1}$)**. When a delegated or an assigned commitment ($C_{i+1}$) discharges, the commitment above it in the chain of commitments ($C_i$) is also discharged. Hence $C_i$ in *pending* state changes to *done*.

The following postulates formally present the state transitions described in Figure 4.6.

**Postulate 5** $\text{create}(C_1) \rightarrow C_1 : \text{active}$

**Postulate 6** $C_i : \text{active} \rightarrow \text{delegate/assign}(C_i) \rightarrow C_i : \text{pending}$ and $C_{i+1} : \text{active}$

**Postulate 7** $C_i : \text{pending} \rightarrow \text{escalate}(C_{i+1}) \rightarrow C_i : \text{active}$

**Postulate 8** $C_i : \text{active} \rightarrow \text{cancel}(C_i) \rightarrow C_i : \text{violated}$

**Postulate 9** $C_i : \text{active} \rightarrow \text{discharge/release}(C_i) \rightarrow C_i : \text{done}$

**Postulate 10** $C_i : \text{pending} \rightarrow \text{release}(C_i) \rightarrow C_i : \text{done}$

**Postulate 11** $C_{i+1} : \text{pending/active} \rightarrow C_i : \text{done} \rightarrow C_{i+1} : \text{done}$

**Postulate 12** $C_{i-1} : \text{pending} \rightarrow C_i : \text{done} \rightarrow C_{i-1} : \text{done}$

**Chain of Orgs.** In addition to the chain of commitments, we see a corresponding chain of Orgs being created. For example, in the current scenario, we have the chain of Orgs:
In this hierarchical chain of Orgs, we observe that the escalates can be easily handled. Any escalation caused lower in the chain can be locally resolved or propagated up. An important observation here is that the path of escalations reverses the chain of Orgs. However, the path of escalation may bypass some of the Orgs on the chain.

4.5.5 Life Cycle of Orgs

We consider an institution as a kind of an Org that may have a longer life time than its member Orgs. An institution continues to exist even if it has no instantiated Orgs. The institution itself may dissolve due to factors such as change or inapplicability of its policies, especially if its entry policies are no longer viable within the institution to which this institution belongs. An Org may dissolve when all of its commitments are done or violated. An institution may dissolve an Org within it if it violates any of the institution’s policies.

We consider four states of an Org, starting from where an Org template is instantiated, resulting in an Org in a working state, until the working Org is dissolved. Assuming an institution already exists, we elaborate on the state transitions in an Org, which is created for enacting its contract. Figure 4.8 illustrates the state transitions in an Org.

- An Org template is instantiated within an institution to form a working Org, which forms the context of the commitments specified in the contract. A working Org has
active commitments.

- A *working* Org *completes* when the associated service engagement completes, i.e., when all of its commitments are *done*.

- A *working* Org becomes *failed* when an Org is unable to resolve any escalate received because of a violated commitment.

- A *failed* Org may be revived back to being *working* by a higher level Org or the institution in which it occurs that handles any escalates. This is made possible either by compensatory actions such as forming new commitments, or by replacing those members responsible for the violation with new members.

- A *complete* Org can be *dissolved*, with all of its commitments *done* and hence removed. A *failed* Org may be *dissolved*, when it cannot be revived back to being *working*. A *working* Org may sometimes be directly *dissolved* by the institution, if it decides based on its policies.

### 4.6 Evaluation

Our policy architecture emphasizes the proactive aspects of policy governance. Monitoring events is crucial here. This section motivates a formal analysis of how agents’ actions comply with policy. Hierarchical policy enforcement may look simple, but in our approach we enable it by generating hierarchical Orgs flexibly and with potential overlaps. Our policy architecture handles hierarchical relationships within an Org such as a real-life organization, as well as peer-to-peer relationships resulting from cross-organizational service engagements.

#### 4.6.1 Challenges: Preemptive Scheduling

Figure 4.9 revisits the scenario of Section 1.4.2. When an ancestor Org receives an escalate of a commitment from a descendant, the ancestor Org can check if it is legitimate, and perform any compensatory action. If it is unable to satisfy the escalate, it can forward the escalate up in the hierarchy. In the present case, National Labs preempts ORNL’s other service engagements (not shown in the figure), and delegates the NHC commitment to ORNL.
Figure 4.9: A preemptive scheduling scenario

Figure 4.10: A messaging sequence diagram for the preemptive scheduling scenario
Figure 4.10 describes how our agent policy architecture handles this scenario. It shows a messaging sequence diagram indicating the communicative actions and their processing by the different components of the architecture.

4.6.2 Handling Conflicts and Escalations

An agent can potentially form multiple commitments with different agents. An agent’s commitments may potentially conflict because of timing or resource constraints. For example, Figure 4.11 shows a scenario where ANL is simultaneously committed to both NHC and the data mining department (DM) of Visa. ANL denies NHC’s request for additional resources because of the two conflicting commitments.

Our policy architecture supports resolution of such conflicts with the help of escalations and a dynamically created Org structure. An agent involved in multiple simultaneous commitments may assign preemptive priorities to selected commitments. In the above example, the conflict between the two commitments created by ANL can be detected via a logic engine. ANL might have a policy to handle such conflicts by canceling one of its commitments on the basis of the specified priorities. More importantly, such priorities would be specified by a higher-level Org when the service engagement is initially created. Priority assignments may depend on the order in which the agreements are created. For example, if NOAA-NL agreement is created first, NOAA may receive the highest priority for hurricane and earthquake modeling, while Visa may receive the best possible priority for data mining.

In the above example, depending on the priorities specified for the service agreements (NL-NOAA and NL-VISA), when NL receives a request to handle an escalate from one of its higher Orgs (Org-NOAA-NL or Org-VISA-NL), NL can either release ANL from NHC or DM and delegate to ORNL, or can release ANL from DM, and put it on hold.

Privacy constraints sometimes limit the handling of escalates directly by the lower Orgs. For example, Org-ANL-NHC cannot resolve NHC’s escalate, because ANL may not share any information about its Visa contract while denying NHC’s request. Hence escalates may be forwarded up the hierarchy until one of the Orgs can resolve it.

**Org Creation Path versus Escalation Path.** We observe that the path of the escalations reverses the path in which the Orgs are created. However, the creation path could be longer than the escalation path. As a case in point, once a commitment is assigned, the assigner may not want to hear if it does not work out. For example, when a bank $B_1$ sells a
loan to $B_2$, and $B_2$ cannot collect on it, then $B_2$ has to deal with the borrower on its own, not through $B_1$.

We formally study the enactment of the various concepts that were introduced earlier in Chapter 2 and Chapter 4. We provide a theoretical framework for enactment of policy compliance in an Org. Key properties of an Org are identified and formally presented.

### 4.6.3 Formal Results on Chain of Commitments in Orgs

We present formal results on state transitions in a chain of commitments. We present some important properties of Orgs with respect to the chain of commitments appearing in them.

**Early release of commitments.** Sometimes, a commitment may be dropped due to various reasons (legal, business, or economic). For example, a commitment may be deemed illegal by a higher level Org or a policy violation may happen. Under such situations, the creditors of the commitment or the higher-level Org may release the debtors of the commitment. The following theorem generalizes the effect of release on the lower commitments in the chain.

**Theorem 1** If $C_i \in H_n$ is released, then $H_n$ completes with all commitments subsequent to $C_i$ that are not previously violated being done.
Proof sketch. From Definition 13 and Postulates 9, 10, and 11.

For example, in the scenario illustrated by Figure 4.7, if the commitment $C_1$ is deemed invalid by Org $O_1$ due to an institutional policy of $M$ that restricts the auction of the item being sold by ABC, $O_1$ may release the commitment $C_1$. Now the subsequent commitments ($C_2$ and $C_3$) in the chain are released and hence become done.

Successful discharge of delegated commitments. The motivation for the following theorem is that, when an agent discharges a delegated or an assigned commitment, it solves the requirements of the higher-level Org that had transferred the responsibility to the agent. Hence the corresponding top-level commitment is also discharged.

Theorem 2 If a commitment $C_i \in H_n$ is done, then all $C_j \in \{C_1, \ldots, C_{i-1}\}$ are done.

Proof sketch. It follows from Postulate 12.

For example, in the example of Figure 4.7, if the DHL-USA discharges the delegated commitment $C_3$ by shipping the auction item to Alice, the commitments $C_2$ and $C_1$ are also discharged and hence become done.

Propagation of Escalations. The following theorem combines the above situations, considering the cases of violations in the delegated or assigned commitments among the lower-level Orgs. An escalation propagates up bringing back the upper commitment to active. An escalate is forwarded up until an agent can potentially discharge the commitment. If no agent can discharge the commitment, the first commitment in the chain is violated.

Theorem 3 If a commitment $C_i \in H_n \ (i < n)$ is done, then each $C_j \in \{C_{i+1}, \ldots, C_n\}$ is done or violated.

Proof sketch. This follows from the state transitions defined by our life cycle model. If $C_{i+1}$ is violated, then $C_i$ would have become active and then become done (Postulate 7). We use the fact that only an active commitment can ever become violated (Postulate 8). Hence, $C_{i+1}$ being violated implies each $C_k \in \{C_{i+2}, \ldots, C_n\}$ is violated. If $C_{i+1}$ is previously done, then $C_i$ is also done (Postulate 12).

In Figure 4.7, if $C_1$ is done, that implies $C_2$ and $C_3$ are either done or violated. If both $C_2$ and $C_3$ are violated, then ABC would have directly discharged $C_1$. But if either of $C_2$ or $C_3$ is done, then that would have made $C_1$ done as well.

Ensuring Completeness and Soundness in Orgs. The life cycle of an Org depends
on the enactment of commitments formed within the Org. Completeness of an Org implies that the enactment of a service engagement arising within it terminates. An Org reaches a complete state only when the commitments created within it are done. An Org is in a failed state when a commitment which has the Org as its context is violated. Ensuring that the chains of commitments are finite is a necessary condition for the completeness of Orgs. One way to ensure the above is to not allow for any delegates and assigns to happen to (1) a higher Org (one of the delegators or assigners), or (2) an escalating debtor or creditor, respectively. However, the above conditions do not guarantee that an Org will reach a complete state because a commitment that does not get redelegated or reassigned in the above manner can still get violated and the Org may not always be able to resolve a resulting escalate.

The correct treatment of state transitions and the existence of necessary policies to take care of any violations ensures the soundness of Orgs arising within institutions. A sound Org ensures that each action undertaken by a member agent does not violate any of the institutional policies.

Postulate 13 A working Org has at least one chain of commitments $H_n$ that is active.

Postulate 14 In a complete Org, all the chains of commitments are complete.

Postulate 15 A failed Org has at least one chain of commitments $H_n$, where $C_1 \in H_n$ is violated.

Postulate 16 In a sound Org, every state transition occurring in every chain of commitments is compliant with the policies of the institution, within which it is instantiated.

Postulate 17 A sound Org that includes chains of commitments with at least one delegation or assignment of the original commitments should necessarily specify escalation handling policies that handle escalates received from delegated or assigned commitments.
4.6.4 Formalizing Policy Compliance in an Org

Policy compliance is required at all levels in a nested Org architecture. Each agent applies its policies to act – respond to or produce communicative acts. We use the example described in Section 1.4.2 of a contract formed between DOC and DOE. Figure 4.12 shows an Org formed with the members: DOC and DOE. DOC is an Org agent with a member NOAA and NOAA is an Org with a member NHC. DOE is an Org with member NL, and NL is an Org with members ANL and ORNL. In this scenario, we observe that a contract is formed at the top level in the Org hierarchy between DOC and DOE, while the actual enactment happens between NHC and ANL. The policies of the agents DOC, DOE, NHC, and ANL are shown in the figure. We introduce some important concepts in the following subsections that build our policy compliance approach.

![Figure 4.12: An example Org](image)

4.6.5 Point of Enactment

When a commitment is formed between two agents within the scope of a Org, the debtor of this commitment may delegate it to other agents in the Org, who can be the debtor’s members. The delegation can continue down the Org structure. The point of enactment is the final debtor that directly discharges the last commitment in the chain of commitments.
**Definition 15** A point of enactment $D_F$ of a commitment $C$ with a discharge condition $F$, is a debtor that brings about $F$ and discharges the final commitment in a chain of commitments, which has $C$ as its first commitment.

In Figure 4.12, the two rectangles with ellipses referring to NHC and ANL are the two points of enactment, for the two commitments formed between NHC and ANL.

### 4.6.6 Delegation Path

A *delegation path* of a commitment begins from its debtor going down the Org hierarchy in a series of delegations till the point of enactment. In essence, the commitments are propagated from the Org all the way to the point of enactment.

**Definition 16** A delegation path ($dp_i$) for a commitment $C_i$ formed within the context of an Org is a list of debtors representing the path of delegations, starting from the debtor who created the original commitment down the Org hierarchy in a series of delegations till a point of enactment $D_F$.

In Figure 4.12, the paths marked with thick arrows starting at the debtors DOC and DOE, and ending at the points of enactment are the two delegation paths. One path belongs to the commitment formed by DOC; the other that of DOE.

Let $DP_O$ be the set of all the delegation paths $dp_i$ for all the commitments $C_i$ of the Org $O$.

### 4.6.7 Assignment Path

Similar to the delegation path, an *assignment path* of a commitment begins from its creditor, going down a series of creditors because of the successive assign operations performed on the commitment by the creditors, till the final creditor that directly benefits from the commitment.
Definition 17 An assignment path \((ap_i)\) for a commitment \(C_i\) formed within the context of an Org is a list of creditors representing the path of assigns, the starting point being the creditor of \(C_i\), down the Org hierarchy till the last creditor who directly benefits from the commitment.

Let \(AP_O\) be the set of all the assignment paths \(ap_i\) for all the commitments \(C_i\) of the Org \(O\).

4.6.8 Contract Formation and Org Enactment

We described earlier in Section 4.5.1 about the creation of Orgs within an institution. A contract is formed within the context of an Org that is formed specifically for carrying out this contract. An Org template with the relevant roles is instantiated for the purpose of this contract to form an Org. The goals of the contract are included in the goals of this Org. The contract terms and conditions are specified as the commitments appearing within the Org, and some conditions can be specified as the policies of the Org and the individual contracting agents. The contracting agents can delegate their commitments to any other agents in the Org, including their own members. Sometimes, the points of enactment can be directly responsible for the formation of a new contract and hence a new Org. In Figure 4.12, a contract is formed between DOC and DOE, both forming members of the common Org \(O_1\). However, the commitments formed between DOC and DOE are delegated to the respective points of enactment NHC and ANL.

The enactment of an Org refers to the successful discharge of the member commitments in compliance with the policies defined by the Org and its members at different levels. We described the life cycle of an Org in Section 4.5.5. An Org lasts till all the commitments are “done.” In our policy-based governance architecture, the commitments are successfully enacted, only if the policies defined by the Org, and its member agents are satisfied. Any operation on commitments is conveyed by a communicative action and these actions are controlled by the policies in our framework. So the successful discharge of commitments and the completion of contracts implies policy compliance in the Org. Hence commitments form a basis for checking compliance in our framework.
4.6.9 Coalitions

A coalition is formed when agents form a contract to achieve certain goals. It includes all the contracting members who form commitments as well as all the agents involved in the delegation paths and the assignment paths till the points of enactment. Coalitions exclude the context Org (within which the given contract is formed), so that it may serve as an (neutral) organizational context for the contract.

**Definition 18** A coalition $L_O$ is a set of agents, within the scope of an Org $O$ within which a contract is formed, who are involved as debtors or creditors in all the commitments formed by the contract. It is the union of all the agents in the delegation paths ($dp_i \in DP_O$) and the assignment paths ($ap_i \in AP_O$) except the context Org.

In the DOC-DOE Org, the coalition for the contract formed between DOC and DOE is a set that includes \{DOC, NOAA, NHC, DOE, NL, ANL\}.

**Definition 19** A path $dp_i \in DP_O$ is compliant if the successful discharge of the corresponding commitment (by the point of enactment in the path $dp_i$), satisfies the policies of all the agents in the delegation path $dp_i$.

In Figure 4.12, path $dp_C$: (DOE → NL → ANL) is compliant, if the commitment to provide grid services is discharged with the conditions that “best” QoS is provided, and that ANL is not under any conflict. Path $dp_C$ is compliant because the policies at all levels are satisfied. However, if a path existed to ORNL instead of ANL, and if ORNL cannot provide “best” QoS, then that path would not have been policy compliant.

**Definition 20** A path $ap_i \in AP_O$ is compliant if the successful discharge of the corresponding commitment by its point of enactment, satisfies the policies of all the agents in the assignment path $ap_i$.

The enactment of a goal in an Org requires the coalition members to have policy-compliant interactions to accomplish the goal. And, especially, if there is a hierarchical
structure, then the policies at all levels higher than the point of enactment should be satisfied. The following theorem applies the compliance of delegation and assignment paths and shows the completion of an Org resulting from a contract formed by a coalition.

**Theorem 4** An Org $O$ resulting from a contract created by a coalition $L_O$ is completed if each path in the sets $DP_O$ and $AP_O$ is compliant.

**Proof sketch:** Follows from Definitions 18, 19, and 20.

In our DOC–DOE example, the contract formed within the Org $O_1$ to provide grid services to NHC is accomplished by the coalition containing DOC, DOE, and others, provided both the delegation paths in the coalition are compliant. In this example, we have not considered any assignment paths. This would be true and the theorem would apply if the associated commitments are discharged if the “best” QoS was ensured for the service. Because of a resulting noncompliant path, the theorem would not apply if ORNL were included in the coalition.

### 4.6.10 Actions and Policy Compliance

We described above some results on policy compliance considering delegations and assignments of commitments formed within Orgs. We generalize and provide improved results on the compliance and completion of an action, based on the different kinds of policies in effect and the events observed. Below $P$ is a policy, $a$ is an action, and $\models$ means entailment.

**Definition 21** $P \models_F a$ means that logical rules $P$ require action $a$ given the facts $F$.

$P, a \not\models_F \text{false}$ means that action $a$ is consistent with (permissible based on) the set of rules $P$ given the facts $F$. $P, a \models_F \text{false}$ means that action $a$ is prohibited by the set of rules $P$, given the facts $F$.

An action $a(A)$ performed by agent $A$ is policy compliant if $a(A)$ complies with the policies, events, and facts stored by $A ((P_H, P_L, E_h, F_C))$. $F_C$ is a set of configuration facts, and $E_h$ is the set of events in the history. Below $F_C$ and $E_h$ are included in the facts $F$ included in the definitions and postulates.
Postulate 18 states that an action must be done if it is required by the hierarchical policies of an agent. Agents in Orgs are not completely autonomous. Hierarchical policies have a higher precedence than local policies. Using Property 1, we see that (when \( P_H \) requires action \( a \)), even if action \( a \) is not permissible with the local policies of the agent, \( a \) must be done, because the hierarchical policies override the local policies.

**Postulate 18** If \( P_H \models_F a \), then \( A \) must do \( a \)

Postulate 19 states that an action must not be performed if it is not permissible by the hierarchical policies.

**Postulate 19** If \( P_H, a \models_F \text{false} \) (\( P_H \) prohibits \( a \)), then \( A \) must not do \( a \)

Postulate 20 states that when \( P_H \) permits \( a \), \( a \) must be performed if required by the local policies.

**Postulate 20** If \( P_H, a \not\models_F \text{false} \) (\( P_H \) permits \( a \)) and if \( P_L \models_F a \), then \( A \) must do \( a \)

Postulate 21 states that action \( a \) must not be performed when not permissible by the local policies, and not required by \( P_H \).

**Postulate 21** If \( P_H \not\models_F a \) (\( P_H \) does not require \( a \)) and if \( P_L, a \models_F \text{false} \) (\( P_L \) prohibits \( a \)), then \( A \) must not do \( a \)

Postulate 22 states that when \( P_H \) and \( P_L \) both permit \( a \), but not require \( a \), then \( a \) may be performed.

**Postulate 22** If \( P_H \) and \( P_L \) do not require \( a \) and if \( P_H, P_L, a \not\models_F \text{false} \) (\( P_H \) and \( P_L \) both permit \( a \)) \( A \) may do \( a \)

The above postulates group the hierarchical policies of an agent which is a collection of the forwarded policies of its ancestors (\( P_H = P_F(A_{\text{root}}) \cup \ldots \cup P_F(A_{\text{parent}}) \)). In \( P_H \), the forwarded policies of the root agent \( A_{\text{root}} \) override those of its child and so on.

**Definition 22** An action \( a \) of an agent \( A \) is said to be policy compliant if it is not prohibited by \( A \)'s policies when it occurs.
**Action Path.** An action path in an Org is a sequence of operations that occur within its scope, i.e., across its descendants, in enacting a service agreement. The configuration administered by the Org agent evolves due to these operations, as the relationships among the administered parties change. An action path for a commitment is a sequence of operation that begins from its create, may proceed with a series of delegates or assigns, and ends with its discharge, cancel, or release. Action paths for other kinds of configuration facts are similar.

**Definition 23** A commitment action path is a finite sequence of tuples \( \langle a_i, A_i, C_i \rangle \) (\( i \) ranges from 0 to \( n \)), each representing a manipulation of the commitment \( C_i \). Here \( a_0 \) is the create of \( C_0 \) by agent \( A_0 \), \( C_{i+1} \) results from action \( a_i \) applied on \( C_i \), and \( a_n \) is either discharge, cancel, or release.

For example, in the NHC–ANL scenario, if ANL discharges the commitment by itself, then the action path is \( \langle \text{create}(ANL, NHC, C_0), ANL, C \rangle \) and \( \langle \text{discharge}(ANL, NHC, C), ANL, C \rangle \).

**Definition 24** An action path is compliant if each of its actions is policy compliant when it occurs.

The following theorem establishes that all action paths in the proposed architecture are compliant.

**Theorem 5** If an action path forms in an Org, it must be compliant.

**Proof:** In an Org, the hierarchical policies are aggregated with the local policies of an agent. We know from Postulates 18 and 20 that any action required by the hierarchical policies of an agent, or required by the local policies of an agent and permissible by the hierarchical policies, must be performed by the agent. At other times, when \( P_H \) or \( P_L \) prohibit an action, then that action must not be performed (Postulates 19 and 21). An agent may perform an action otherwise when both \( P_H \) and \( P_L \) permit it (Postulate 22). Hence all actions taken by the agents in an Org are permissible both by their local and hierarchical policies and are policy compliant by Definition 22. Thus the theorem follows from Definition 24.
Definition 25 An Org is complete for a configuration fact when an action path for that fact exists within that Org. An Org is complete if it is complete for all relevant configuration facts.

Potentially, an Org could be incomplete if no compliant actions were to exist under some circumstances. However, our definition of compliant action paths includes those that end in discharge, cancel, or release. By thus expanding the possibilities for the final action of an action path, we create more opportunities for an action path to exist. Further, our architecture prioritizes upper-level policies over lower-level policies. But an architecture cannot by itself ensure completeness.

For example, it is possible for the descendants of an Org to interfere with each other. In general, it is possible for two descendants of an Org (neither of which is an ancestor of the other) to have mutually conflicting policies. Barring such interference, an Org can complete for a configuration fact although the configuration may be violated externally. Theorem 6 captures this intuition.

Definition 26 Two actions are independent if either can occur and if both occur, they can occur in either order with the same resulting state.

Theorem 6 Completeness of an Org is ensured if all pairs of actions of all pairs of descendants (neither of which is an ancestor of the other) are independent.

The above notion of completeness does not entail a “happy” ending. For example, NL may complete the commitments it delegates internally by releasing all of them, although that might cause NL to cancel its commitment to NOAA. A commitment that is not discharged, delegated, assigned, released, it is considered canceled. To ensure completeness of Orgs, delegates and assigns can happen only “downwards”. We have to ensure that there are no cycles and no infinite chains of delegates or assigns. Thus an Org may potentially become “complete for a commitment” provided no commitment is ever redelegated or reassigned to an escalating party. Ultimately, completeness resulting in all commitments being discharged depends upon the various parties having the right policies for the circumstances in question. In particular, the cancel or release of a commitment may cause the creation of
other commitments. The design of such policies is an important topic beyond the current scope of this work.

4.7 Evaluation by Prototyping Practical Scenarios

We have implemented a prototype system called MAVOS that has a policy architecture based on Jess and conventional messaging middleware. This prototype implements a multi-agent architecture for Orgs and demonstrates the policy-based enactment of commitments. The prototype has a simulator which instantiates multiple agents and Orgs, their policies and the members. Agent interactions using communicative actions is supported and the agent policies control these communicative actions. We have implemented the example scenario discussed earlier and demonstrated it using our prototype. Each agent (including Org agents) has a separate policy engine which is applied to respond or send communicative actions and also perform other domain specific actions.

The significance of our approach can be seen by the wide practical applicability of its outcomes. Our approach provides innovations in policy architecture to better model and manage complex Orgs. The practical applications of this approach include,

- A direct application is seen in modeling provisioning of computational resources for large scientific grid environments and Orgs formed thereby, but applied across Orgs and managed in the face of dynamic events. For example, modeling interoperation of production grids, which is fast becoming an important area.

- Practical business scenarios involving contracts, subcontracts, nested governance architectures, and business processes spread over multiple organizations form ideal candidates for the proposed approach. For example, business process outsourcing (BPO) involves the refactoring of an enterprise’s processes so that external parties can be engaged to gain improvements including efficiency and reliability. Our policy-based governance approach can better model these BPOs and their dynamic events. A supply chain scenario can also be nicely modeled using our multiagent conceptual model and managed using our governance architecture.

- Governance in a practical business scenario involves provisioning of services, where resources controlled by multiple organizations come together, and sophisticated service
level agreements (SLAs) are required. These SLAs can be modeled using the contracts and commitments in our approach.

We consider specific use cases and demonstrate the practical applicability of our approach in the following subsections.

4.7.1 Grid Use Case Scenarios

We consider the use case scenarios studied for the Open Grid Services Architecture (OGSA) [Foster et al., 2004]. These use cases include topics of relevance to both e-science and e-business. We demonstrate below, how our approach provides a high-level modeling for these use cases.

Commercial Data Centers Use Case

This use case considers the challenge of integrating IT resources such as servers and storage into data centers in order to reduce the ownership costs. Many enterprises outsource their IT resources and their management to commercial data centers. This scenario describes how OGSA provides a grid-based solution for a set of functional requirements. Our approach provides a high-level modeling to this use case scenario and our policy-based architecture supports handling the key functional requirements. We apply our approach and elaborate on this scenario below.

The roles involved in this scenario include the following: Grid administrator (GA), IT system integrator (SI), IT business activity manager (AM), and an End user (EU). In our approach, these roles are instantiated by agents that interact with each other, participating in several service engagements. The GA agent administers the commercial data center, and hence can be considered as the grid service provider. An SI agent can use the data center’s grid services to satisfy adequate QoS requirements while building distributed and heterogeneous systems. The AM agent uses the grid to provide different business services to end user (EU) agents.

This use case considers an example scenario where the grid or the data center implementing the grid provides a solution to multiple in-house systems. Current in-house systems are mostly isolated. For example, personnel management system includes various
systems such as finance and accounting, order receiving, and customer relationship management that are mostly isolated, where each in-house system runs on its own IT resources and is prepared for high availability and increased workload. Workload keeps varying and peaks do not occur at the same time, which leads to several idle IT resources. In such scenarios, the data center solution leads to better IT utilization and reduces ownership costs.

We model this example using our approach. Here, the GA agent administering the grid, is provisioning grid services to each in-house system’s business activity manager. GA agent provisions grid services and provides functions such as advance reservation, allocating resources, autonomous management including failover and so on to the AM agent. Also, multiple GA agents belonging to different remote data centers could collaborate to improve scalability and availability.

Our agent-based approach enables proper modeling of the relationships, interactions among the different roles. The AM agent provides different services to end users and these services are represented as commitments from an AM agent to the end user agents to provide different services. The service engagement between the data center and a business activity manager can be represented as a commitment from a GA agent to an AM agent. In the OGSA solution, the service engagements between AM agents and the end users are created within VOs created in the grid, because VOs offer the required QoS and are secure, reliable, and scalable. In our approach an Org forms the context of the commitments created. The AM agent can be involved with different remote data centers simultaneously. Hence a VO can include multiple remote data centers.

Functional requirements. The data center can simultaneously engage with multiple enterprises to provide grid services. Monitoring for specific events and usage logging are essential requirements in such scenarios. Managing resource utilization, scheduling, fault handling, security, load balancing and disaster recovery suggest policy-based mechanisms.

The functional requirements considered in the use case include discovery, authentication, authorization, accounting, advance reservation, brokering, data sharing, provisioning, scheduling, fault handling, policy, security, handling virtual organizations, monitoring, load balancing, and disaster recovery. The OGSA solution provides a set of grid services to handle the above functional requirements. Our approach focuses on functions where policies apply and it provides a policy-based architecture to handle such functional requirements.

We list below the OGSA services that support the functional requirements listed
above that relevant to this use case. Our approach provides policy-based solutions that apply to most of the services below.

1. **Name resolution and discovery service:** Discovery services enable the participating roles to select a particular CDC that they will be using. Our approach does not consider discovery, as we model a service engagement where the participating roles are already decided.

2. **Security service:** This service enables authentication, authorization, and accounting functionality. The grid authenticates and authorizes the job requests submitted by different customers. The policies are checked if a customer has the right to perform the job request and resource access control policies. Our approach assumes an underlying security framework that can handle authentication and authorization, but allows for the agents participating in a service engagement to specify policies such as access control policies or any other policies about an agent’s security preferences.

3. **Reservation service:** The advance reservation functionality is enabled by this service. In our approach, we can employ the reservation service to trigger the creation of commitments according to the reservation made. Policies can be written to determine the actions to take if there are conflicting reservations.

4. **Brokering service:** enables brokering of resources. In our approach, if the resources are limited, certain commitments can be canceled. Basic solutions for brokering such as prioritized resource allocation and advance reservations can be handled in our approach by assigning priorities for commitments and having policies that control their enactment.

5. **Data management service:** handles data sharing within a data center and across multiple data centers. This service is also required for disaster recovery of data from other data centers. Our approach does not handle any low-level data management services, but we can support data accessibility policies for the different roles involved in a service engagement. We can also control data accessibility during delegation or assignment of commitments with the help of adequate policies.

6. **Provisioning and resource management service:** These services handle the provisioning of resources, both application and data in the grid. This service is also used for creating
VOs spanning across multiple data centers. Our approach models service engagements at a high-level and supports handling of provisioning and resources management with the aid of policies that govern the agent actions. These actions could include domain actions such as deploying an application in the grid.

7. **Scheduling service:** enables scheduling of jobs based on priority. Several underlying scheduling algorithms are provided by the grid solution. Our approach assumes an existing scheduling framework, but assists an agent to make scheduling decisions based on the priorities of commitments that can specified as priority policies.

8. **Metering and accounting service:** is used to meter the resource usage and provide accounting. Our approach assumes the availability of a metering and accounting service, but can support enactment of certain policies based on the information supplied by the metering and accounting service. The information supplied by the metering and accounting services are caught as events by PMP in our agent policy architecture.

9. **Fault handling service:** enables fault notification and autonomous management when a job does not complete successfully or has an error with the help of fault management policies. This is also used for disaster recovery, when the jobs are relocated to other available data centers. Our approach supports fault handling by modeling faults as cancellation of commitments. Policies can trigger compensating action scenarios such as reallocation, preemption, or creation of new commitments.

10. **Policy service:** enables the functionality of various kinds of policies such as brokering policy, event-based policies. Our policy-based architecture handles enactment of commitments and supports the various examples relevant to this use case scenario.

11. **Monitoring service:** enables monitoring of an application running in the data center. Our policy architecture has the PMP module which continuously monitors for any events relevant to the agent and these events may trigger policies defined by the agent.

12. **Deployment service:** is used along with the provisioning service to deploy applications and data in the data center. Our approach assumes the availability of an appropriate deployment service and provides a high-level model of provisioning of resources in the grid.
Org design patterns that were described in Chapter 3 are applicable in this use case scenario. We illustrate the applicability of delegation, escalation, preemption, and reallocation patterns below. Other patterns applicable in this scenario are creation, cancellation, release, and discharge.

Delegation. A commitment from an AM agent to an end user to provide certain grid services is delegated to a GA agent who administers the data center and provides the grid services. The pattern instantiation is represented as follows: Delegation(AM, EU, GA). Figure 4.13(a) illustrates the delegation pattern applied to this scenario. \( C \) is the commitment from AM to EU to provide certain services and \( C_{Del} \) is the delegated commitment from GA to EU. \( C' \) is the commitment from GA to AM to accept any delegations for providing the grid services. \( P_1, P_2, \) and \( P_3 \) are the policy points of AM, GA, and EU respectively, and are instantiated by policies relevant to this scenario. For example, AM might have a policy that selects a particular data center to delegate a certain commitment with an end user.

Escalation. When service engagements are created between an AM agent and an EU agent, several service-level agreements (SLAs) about certain QoS requirements are specified in the contract. The commitment from AM to EU can capture such SLAs. If any of the SLAs are violated, an EU agent can escalate the commitments to the context Orgs. The pattern instantiation is specified as follows: Escalation(AM, EU). Figure 4.13(b) shows the escalation pattern instantiated with roles from this scenario.

Preemption. Preemption is applicable in scenarios when there are multiple jobs and VOs created with limited resources available at the data center. An AM agent creating commitments with different end users may specify varied priorities depending on the urgency of the...
jobs. Under cases of conflict, an AM agent may preempt a GA agent from a lower priority commitment to ensure a higher priority job to be discharged. This pattern instantiation is represented as follows: \textit{Preemption}(GA, EU-L, EU-H, AM). Figure 4.14 (a) illustrates the application of preemption pattern to this scenario. Here AM is the preempering role who can preempt GA from a lower priority commitment $C_L$ to an end user EU-L.

\textbf{Reallocation.} Reallocation pattern is applicable in this use case scenario, when a business activity manager has created a VO that includes multiple data centers. If a commitment from an AM agent to an EU agent is delegated to a GA agent in one of the data centers and if that data center is overloaded or has a failure, then the commitment has to be redelegated to another data center. This reallocation pattern is instantiated as follows: \textit{Reallocation}(AM, EU, GA−1, GA−2), where GA−1 is the data center that fails to provides services to the end user, and GA−2 is the compensating data center. Figure 4.14 (b) illustrates the reallocation pattern for this scenario, where the commitment $C$ is redelegated to GA−2.

\textbf{Severe Storm Modeling}

This use case scenario involves a consortium of meteorologists and environment modelers who create VOs in a grid to enable them to make accurate predictions of exact locations of severe storms. They require real-time weather instrumentation and large scale simulation coupled with data modeling. For reasons of brevity, we do not elaborate on this and rest of the use cases. Our approach provides a high-level conceptual model of this scenario.
and can handle most of the requirements presented. Preemptive scheduling is applicable in this scenario and is supported by our approach. If certain events monitored suggest a forthcoming storm, the grid might have to be preempted from other lower priority resource allocations, and the grid has to provision the required resources for storm modeling.

**Online Media and Entertainment**

This use case scenario involves the following roles: a consumer and a developer who consume the entertainment content, a service provider that hosts the entertainment content, and a publisher that offers the entertainment content. The key requirement here is to dynamically manage resources based on workload demands and current system configurations. Interactive entertainment such as online games involves many interaction scenarios between the roles and it is important to guarantee response time and other QoS guarantees. OGSA provides a grid data center solution with different services to handle this scenario. Our approach provides an accurate modeling of the interactions among the roles participating in the service engagements. Policies can handle key functional requirements such as resource management, billing, failure handling, provisioning management, and workload management. Assuming the availability of the underlying low-level services and application to enable online media and entertainment, our approach provides a high-level modeling of the interactions and can enable policy-based governance decisions of each of the autonomous participants.

**National Fusion Collaboratory**

This use case scenario presents a “network services model” where software generated for fusion research and a set of supporting platforms are provided by a service provider to be remotely accessed by clients. This removes the efforts of the individual users to maintain software versions, debug incompatibilities and other maintenance costs. This use case scenario involves the roles of a consumer who is a fusion scientist, and a service provider who hosts the software and the platform. The key challenges and requirements addressed in this use case scenario include: QoS-based execution to handle end-to-end QoS guarantees, availability contract to manage resource utilization when there is a high demand from multiple users, usage policies to control resource access, service execution, flexible delegation
of rights, and community accreditation. The use case includes an example service engagement scenario where a scientist from one NFC site (client) needs to remotely run a software installed and maintained at another NFC site (service provider) within a specified time bound. The scientist enters into a contract with the provider to receive this requested QoS requirement and is guaranteed execution during the availability window. Preemption needs to be supported when there are time-critical requests and the resources are limited at the provider’s site. Our approach can adequately model the roles, interactions, and scenarios involved in this service engagement. Commitments can be created among the roles and their enactment is controlled by policies of each of the roles.

**Services-Based Distributed Query Processing**

This use case scenario involves a services-based solution for the evaluation of queries over one or more existing services. The grid provides several grid data services for clients who require distributed query processing. This use case has a requirement to integrate data from distributed and heterogeneous resources with analysis services. Our approach can model the engagement between the grid data sources, services, and the clients who submit query requests. The execution of queries can be monitored and appropriate policies can be written for handling monitored events. Events monitored provide updates on availability of computational resources and can trigger any reallocation or preemption policies.

**Grid Workflow**

This use case scenario presents workflow as a way to make new services by composing existing services. This use case was presented for the “service orchestration service” of OGSA. Example scenarios considered for the use of workflow include: connecting simple services to create a new grid service, grid job workflow that specifies the execution order, input, output, etc., business process description by connecting several services, and system administration workflows. In our approach, we would model enactment of service engagements by keeping track of the manipulations of the commitments. A workflow can be specified by providing a sequence of communicative actions.
Grid Resource Reseller

This use case involves a reseller of grid resources acting in between the provider or the owner of grid resources and the end users. Resellers may combine resources from different providers and sell as bundles to end users. To satisfy the SLAs with end users, a reseller may switch the providers. This use case scenario involves many sophisticated policies for resource management and accounting of resource usage. A resource reseller may not be able to resell in a way that violates the owner’s policies. Our approach can accurately model this use case scenario. Resellers selling a grid resource create commitments with different end users. A reseller can choose the appropriate resource owner for a particular end user and delegate the commitment. Any violations or exceptions can be handled by the escalations received from the end users.

Inter-Grid

This use case scenario is similar to the commercial data center scenario, but emphasizes the abundance of applications that are not grid enabled and are difficult to change. The high-level modeling of this scenario using our approach is similar to the commercial data center scenario.

Interactive Grid

An interactive grid allows applications to run on a grid with support for individual user interactions. Users will be able to interact and synchronize with processes that have been farmed out or restarted. Interactive grid also allows automated schedules, and automated series of actions, and hence allowing a job to be controlled by an external agent. An agent thus can control its resource needs interactively during runtime. In the high-level modeling of service engagements created as commitments between the end users and the grid service providers, interactions are captured as manipulations of commitments. For example, a grid provider can cancel a commitment to a user when it fails to allocate services to the user.
Grid Lite

This use case scenario is similar to other grid scenarios but emphasizes the application of grid services for small devices such as PDAs, cell phones, firewalls, and so on.

Persistent Archive

This use case presents a scenario of persistent archive that provides several archival processes such as appraisal, accession, arrangement, description, preservation, and access. The target users of this archive include groups that handle large data collections. Persistent archive require multiple site replication for handling scenarios such as disaster recovery, fault tolerance, performance and curation. The OGSA data services support the functional requirements required for persistent archiving. Our approach can provide high-level modeling of data services as commitments between the persistent archive and the user, but assumes existing underlying low-level services to enable archiving.

Mutual Authorization

This use case captures a grid service provisioning scenario similar to the commercial data center, but with an emphasis on the security and protection of intellectual property. Sometimes a job submitted by a user running on a grid resource may get transferred or resubmitted to another grid resource because of load balancing or satisfying requested quality of service. But the user may not have authorized the secondary remote resource. Hence there is a need for a callback mechanism to the user to obtain the required authorization for the secondary resource. The OGSA services handling this use case include name resolution and discovery, policy, events, security, and service orchestration. This use case acts as a best example in our approach to capture the notion of delegation of commitments. A commitment from a grid resource or an agent owning the resource to a user can be delegated to another secondary resource. The proposed approach can model interactions among agents, where the user has to approve the delegation of the original commitment to another resource.
Table 4.1: Use cases and applicable Org patterns

<table>
<thead>
<tr>
<th>Use Case</th>
<th>De</th>
<th>Es</th>
<th>Pr</th>
<th>Ra</th>
<th>Cr, Re, Ca, Di</th>
</tr>
</thead>
<tbody>
<tr>
<td>Commercial Data Centers</td>
<td>Y</td>
<td>Y</td>
<td>Y</td>
<td>Y</td>
<td>Y</td>
</tr>
<tr>
<td>Severe Storm Modeling</td>
<td>Y</td>
<td>Y</td>
<td>Y</td>
<td>Y</td>
<td>Y</td>
</tr>
<tr>
<td>Online Media and Entertainment</td>
<td>Y</td>
<td>Y</td>
<td>Y</td>
<td>Y</td>
<td>Y</td>
</tr>
<tr>
<td>National Fusion Collaboratory</td>
<td>Y</td>
<td>Y</td>
<td>Y</td>
<td>Y</td>
<td>Y</td>
</tr>
<tr>
<td>Services-Based Distributed Q.P.</td>
<td>Y</td>
<td>Y</td>
<td>Y</td>
<td>Y</td>
<td>Y</td>
</tr>
<tr>
<td>Grid Resource Reseller</td>
<td>Y</td>
<td>Y</td>
<td>Y</td>
<td>Y</td>
<td>Y</td>
</tr>
<tr>
<td>Grid Workflow</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>Y</td>
</tr>
<tr>
<td>Inter-Grid</td>
<td>Y</td>
<td>Y</td>
<td>Y</td>
<td>Y</td>
<td>Y</td>
</tr>
<tr>
<td>Interactive Grid</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>Y</td>
</tr>
<tr>
<td>Grid Lite</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>Y</td>
</tr>
<tr>
<td>Persistent Archive</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>Y</td>
</tr>
<tr>
<td>Mutual Authorization</td>
<td>Y</td>
<td>Y</td>
<td>Y</td>
<td>Y</td>
<td>Y</td>
</tr>
<tr>
<td>Resource Usage Service</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>Y</td>
</tr>
</tbody>
</table>

**Resource Usage Service**

This is the last use case presented in the OGSA use cases document, and it is regarding one of the core OGSA services. This service facilitates the usage metrics produced by applications, middleware, OSs, and physical resources in a distributed, heterogeneous environment. The key OGSA services relevant in this scenario include metering and accounting, monitoring and policy. Our approach can use a resource usage service, monitor the enactment of commitments, and apply resource usage policies to control the resource utilization.

Table 4.1 summarizes the applicable Org patterns for the above use cases. Here, the first row lists the patterns as: delegation (De), escalation (Es), preemption (Pr), reallocation (Ra), creation (Cr), release (Re), cancellation (Ca), and discharge (Di).

**4.8 Discussion**

Governance recognizes that service engagements involve autonomous parties. Here, the interactions occur across administrative domains and there is no unique locus for policy application. Standard operating procedures apply and the participants can seek recourse through escalations. Our policy-based governance architecture overlays a rich model to express administrative interactions, supports obligations and permissions, and accommodates proactive, not just reactive, behaviors. We evaluated our approach by presenting theoret-
ical results on the compliance and completion of overall behaviors from specified policies, and a prototype was built using a conventional policy engine and messaging middleware. A use case based evaluation was performed to show how our approach can model relevant use cases at a high-level. As future work, we consider theoretically handling specific cases of conflicting service engagements, strengthening the completeness results exploiting the Org structure, and handling potential nontermination scenarios. Applying the prototype on the case studies considered on resource sharing in multiorganizational IT infrastructures is considered as future work. The following subsections discuss some important topics that are relevant here, but detailed analyses of them are considered as future work.

4.8.1 Detecting and Resolving Conflicts in Service Engagements

Service engagements can conflict. It is important to detect and resolve any conflict. We can use a priority-based conflict resolution mechanism. We assign priority values for the commitments to simplify the selection of commitments that are to be canceled or delegated. The priorities may be assigned on the basis of the importance or the benefits of discharge of the commitments. For example, in the supply chain scenario introduced earlier, Goodyear may be committed to several car manufacturers in addition to Ford. If Goodyear fails to discharge all the commitments simultaneously, it may cancel or delegate those commitments that are of minimal profits. Policies of an agent specify the various scenarios of conflicts among commitments. Any conflict among commitments results in an exchange of some messages between the contracting members and their Orgs.

4.8.2 Dependencies among Service Engagements

Contracts formed by a set of agents can sometimes have dependencies on each other. Dependency between contracts is captured by the dependencies among commitments formed within the contracts. Commitments formed within a single contract may also be dependent on each other. For example, the discharge of one commitment can cause the discharge of another commitment. These dependencies among commitments can be captured using conditional commitments, where the precondition of the conditional commitment may specify an operation of another commitment. For example, in our supply chain scenario, Ford commits to a car dealer to supply a certain number of cars. But the discharge of this com-
mitment happens when the commitments of all the individual car component suppliers are discharged. In a real-estate contracts scenario, a contract may be formed between a buyer (say, A₁) and a seller (say, A₂) for A₂ to sell a house to A₁. However, A₁ has a condition that he can buy this house only if he is able to sell his present house to another buyer (say, A₃). Now the contract formed between A₁ and A₂ for A₁ to buy a house from A₂ is dependent on the second contract. Figure 4.15 illustrates this scenario by showing the commitments formed by A₁ to A₂ and A₃. The commitment C₃ is a conditional commitment that has a precondition F₅ which requires the commitment C₄ to be discharged. In other words, C₃ is dependent on C₄. The dependencies formed between commitments can potentially result in deadlocks if they are not detected and resolved. Detecting and resolving such scenarios of dependencies is considered as future work.

4.8.3 Social Relationships Applied in Policies

Social reasoning mechanisms and relationship analysis play an important role in the design of agent architectures, enabling an agent to evaluate and reason about others using its dependencies with others [Sichman and Conte, 2002; Ashri et al., 2005]. The agent relationships can influence the actions taken by them. Relationships among different agents formed over a particular contract or a goal bring them together to form Orgs. In the Org context, relationships can exist between an Org and its children, as well as among the siblings. In an e-commerce setting, a service provider may depend on a consumer, or a consumer may depend on a provider. Dependencies are dynamic, because they can be formed and revoked at run time in the life cycle of an Org. Policies can specify relationships among agents and hence can influence the operations on commitments and contracts. The study of the
effect of agent relationships on contract enactment in a Org setting is very promising. The relationships captured while describing Orgs can dynamically change and becomes crucial for describing Org behaviors.
Chapter 5

Related Work

Some relevant background was discussed in Sections 2.1 and 4.1. We study other relevant work and provide a comparative evaluation of the proposed approach.

5.1 Organizations and Multiagent Systems

An organization is a group of entities that emerges in cooperative environments when different entities work together to achieve local and collective goals. Agents representing individual principals have been applied to model organizations and their members. Organizations are larger scale than individual agents, because they have reasoning, knowledge, and capabilities beyond those of individual agents [Singh and Huhns, 2005, ch. 17]. Multiagent systems are systems composed of several agents with capabilities of reaching goals that are difficult to achieve by an individual agent.

Organizations in multiagent systems have been previously studied by many researchers and different kinds of organizational designs have been suggested to suit different situations. Horling et al. [2005] survey the major organizational paradigms used in multiagent systems. They identify a variety of organizational designs used by multiagent systems and their effect on the performance characteristics of organizations. Some important designs include coalitions, congregations, and teams. Abdallah et al. [2004] study cooperative coalition formation in organizations. They present a distributed algorithm that uses an un-
derlying organization to guide coalition formation. Brooks and Durfee [2003] demonstrate how “congregations” of agents can benefit multiagent systems in searching for other agents and minimizing the number of interactions and search costs. Teams are formed when a number of cooperative agents get together to accomplish a common goal [Tambe, 2003]. Agents coordinate their actions in a way that is consistent and supportive of their team’s goal. Our approach considers Orgs that host service engagements resulting from the contracts and the associated commitments created among autonomous agents. Here, the contracting agents collaborate in the context of a common Org to accomplish the contract goals. Our approach can be thought of as addressing the same basic problem, robust teamwork, but limited to Orgs where contracts capture the essence of the interactions and the commitment operations support responses to various exception conditions. There are various subtleties possible in human organizations, but our approach considers multiagent Orgs from a computational perspective, where the collaborating entities such as individuals or enterprises are modeled as agents.

5.1.1 Organizations and Institutions

Pacheco and Carmo provide a role-based model to specify an organized collective agency [Pacheco and Carmo, 2003], where a group of agents acting together give rise to a new agent, termed the institutionalized agent. This institutionalized agent is characterized by a set of roles, and its behavior is determined by the agent actions. The proposed approach is similar to the way an Org template is instantiated within an institution, but offers a study of the dynamics of Orgs. Cardoso and Oliveira describe institutions that facilitate both the creation and the enforcement of contracts among agents [Cardoso and Oliveira, 2007]. Contracts are formalized using norms, which are used by an institution while monitoring the contracts, thus making the institution a dynamic normative environment. In contrast, our approach considers an institution to host several Orgs that form the contexts of the commitments created among agents.

Normative systems. The existing literature on institutions treats them as normative systems that can be modeled using agent-based organizational structures. Normative systems specify norms for their members and can be modeled using agent-based Org architectures. An agent’s behavior is constrained by the norms set by the organization. Noriega describes an agent-mediated electronic institution as a computational entity that imposes
restrictions on the agents’ behaviors [Noriega, 1997]. A Fish Market is an auction house that is modeled as an agent-mediated institution in [Rodriguez-Aguilar et al., 1998]. Vázquez and López y López [2006] describe agent-based hierarchical organizations modeled using a normative multiagent framework. Vázquez-Salceda et al. propose an organizational model for normative institutions [Vázquez-Salceda et al., 2005]. Their framework models organizational structure, agent interactions, and the normative structure separately from the agents. Boella et al. [2005] propose a conceptual model of virtual organizations as normative multiagent systems. Here, they recursively model normative systems and demonstrate the dynamic aspects of virtual organizations using different types of interactions between the normative systems and the agents who have assigned roles. Lopez et al. provide a model for normative reasoning by agents, which guides their decision making [López y López et al., 2004]. Dastani et al. show how organizational roles and norms influence an agent’s goals [Dastani et al., 2002]. Our approach places organizations within an institution, acting as a normative system, and modeled as special organization. Institutions can specify policies stating the norms for its members. Our approach provides a commitment-based reasoning to the interactions between agents and Orgs. The proposed Org architecture can be applied to a normative system where the policies can specify the norms of a society which eventually form the local policies of an agent.

ISLANDER [Esteva et al., 2002] is a declarative language for specifying electronic institutions. Here, an institution comprises a dialogic framework, scenes, performative structure, and norms. The dialogic framework specifies the valid illocutions, exchanged by the participating roles, respecting a common ontology, knowledge representation and a common communication language. The scene contains a set of agents playing different roles interacting with each other to accomplish a given activity. The performative structure shows the relationships among scenes. The connections among the scenes define what agents may move from one scene to another subject to constraints. The norms include the commitments and rights of the participating agents. In contrast our approach describes an institution as a kind of an Org that typically has longer life cycles than the other Orgs. The Org template specification is comparable to the modeling concepts using in ISLANDER. An Org template specifies the roles and the commitments possible among the roles. Our conceptual model provides a list of communicative actions possible among agents. A scene is comparable to a context Org that encloses a set of agents coming together in a contract. The execution of a contract results in a nested Org structure because of the manipulations of commit-
ments. This connection of Orgs is comparable to the performative structure specified by ISLANDER. Norms can be specified as institutional policies, and agents instantiating an Org template may enter into certain institution specified commitments.

**Life cycle of institutions.** In [Muntaner-Perich and de la Rosa Esteva, 2007] the authors present a 3F life cycle model for dynamic electronic institutions and it includes three phases namely, *Formation*, *Foundation*, and *Fulfillment*. Here, a coalition of agents is formed in the first phase, which is turned into a temporary electronic institution in the second phase, and finally it is dissolved in the final phase. In contrast, our approach provides a life cycle of Orgs that are instantiated and dissolved within an institution, and the institution itself may have a longer life time.

### 5.2 Contracts and Organizations

A contract binds different parties in an agreement specifying certain terms and conditions of a service engagement among the contracting parties.

#### 5.2.1 Electronic Contracts

Electronic contract systems automate manual paper-based contract processes. Karlapalem *et al.* [2001] present ER EC, an entity-relationship (ER) model extended to model electronic contracts, which includes the core constructs of *contracts, clauses, activities, parties*, and *exceptions*. By contrast, our approach develops an agent-based model for enacting service engagements based on contracts. Karlapalem *et al.*’s notion of a clause is comparable to a commitment in our approach, and activities to agent actions. Any exceptions and conditions of a contract can be specified as policies and enforced by the participating agents.

Chieu *et al.* [2007] present an electronic contract management system that supports initiating and terminating contract workflow, tracking different contract life cycle stages, and organizing and controlling access to the various contract documents. By contrast, our approach includes a nested Org model which scopes the service engagements appropriately. The institutional model provides a social and legal environment within which service engagements can be monitored and escalations allow for handling any exceptions.

Xu and Jeusfeld [2003] formalize electronic contracts to representations that en-
able their automatic monitoring. A contract is executed by a series of actions, and contract constraints define certain rules on the order in which a set of actions can occur. Commitments are defined which specify mutual obligations among contracting parties to guarantee that such contract constraints are met. They describe commitment graphs, a graphical tool to demonstrate commitment relationships among the various parties involved in a contract. Using temporal logic and employing guard expressions, they capture any potential contractual violations and specify algorithms to proactively monitor for any violation. On the contrary, our approach models contracts as a set of commitments, where each commitment is modeled as a three-party obligation from a debtor to a creditor within a context Org. The execution of a contract is modeled in our approach using different commitment manipulations such as delegate, cancel, discharge and so on. So the actions considered in our approach include not just domain actions, but also the communicative actions such as the various commitment manipulations. We use a policy-based governance approach to control the actions. Our policy architecture enables handling of violations and supports proactive mechanisms aided by monitoring for events.

5.2.2 Contracts in Organizations

Boella and van der Torre [2004] discuss contracts in the context of multiagent organizations. They formalize contracts as having constitutive rules that demonstrate how the creation of a contract relates to the mental attitudes (beliefs, desires, and goals) of an organization. Dignum et al. describe an organizational model for agent societies using contracts [Dignum et al., 2002]. In our approach, the commitments created among agents can be considered as a contract, and the enactment of the contracts involves various operations on commitments. Commitments are similar to Boella et al.’s [2004] notion of obligations. However, our approach considers various operations on commitments and provides the dynamics of commitments, thus yielding a more precise model of the enactment of service Orgs.
5.3 Roles, Agents and Organizations

Roles have been extensively used to represent organizational structure in agent based systems. Organizational hierarchy represents the relationships that exist between roles. A role is an important abstraction in multiagent systems, where an agent can adopt different roles at different times.

5.3.1 Role-Based Organizational Design

Wooldridge et al. present Gaia as a methodology that provides an organizational understanding of agent systems [Wooldridge et al., 2000]. An organization is viewed as a collection of roles that relate to other roles and take part in interactions. Roles can specify responsibilities, permission, activities, and protocols. The Gaia design stage consists of three models – an agent model, where agents instantiate the roles and roles are aggregated into agent types that form a hierarchy, a services model, where the main services of the agent are identified, and an acquaintance model, where the communication links between agent types are designed. The proposed approach has a richer conceptual model with different design patterns capturing important aspects of service engagements that are modeled using commitments. It has a richer architecture, where agents who instantiate the roles within an Org template, form commitments and contracts within the context of an (instantiated) working Org. Hence, supporting richer nested Org architectures. The acquaintance model described above can be implemented in our approach using the communicative actions among agents and the policies that control them.

5.3.2 Role-Based Management Systems

Roles are commonly used in the management and security of distributed systems. In agent systems, roles have been used to specify organizational structure, distribution of responsibilities and duties, manage obligations and their delegations and to manage access control. Role Based Access Control (RBAC) was introduced first by Ferraiolo and Kuhn has become a predominantly used security mechanism in multi-user and large distributed networked systems [Ferraiolo and Kuhn, 1992]. In RBAC systems permission is associated with a set of pre-defined roles, and the users are made members of appropriate roles. Sandhu et al. provide a novel framework of reference models to systematically describe the various com-
ponents of RBAC, and their interactions [Sandhu et al., 1996]. The base model $RBAC_0$ describes the key components of RBAC namely, $users$, $roles$, $permissions$, $session$, $user assignments (UA)$, and $permission assignments (PA)$. $RBAC_1$ introduces role hierarchies, and $RBAC_2$ introduces constraints among the roles. $RBAC_3$ combines $RBAC_1$ and $RBAC_2$ to provide support for both role hierarchies and constraints. The proposed approach can support such role based access control in the form of agent policies. The policy architecture can enforce access control policies to any resources that are both active and passive, as opposed to RBAC which emphasizes passive resources.

5.4 Policy Models, Languages and Architectures

We present some policy models, languages and architectures that are relevant to our proposed approach. Although there is a lot of work on policies for distributed management, our approach is unique in addressing service engagements involving autonomous, policy-based participants.

5.4.1 Policy Models

DEN-ng Policy Model. The DEN-ng policy model is an object-oriented information model that uses patterns and roles to model policies designed for next generation networked applications and services [Strassner et al., 2007]. DEN-ng defines concepts such as $Policy-Concept$, which models generic concepts related to policy, as the root of the model and which is contained in $PolicyDomain$. A $PolicyDomain$ is a collection of entities and services that are governed using policies and contains other $PolicyDomains$. This policy model defines four types of deontic policies (authorization, obligation, prohibition, and exemption) and two types of metapolicies (delegation and revocation). Our conceptual model of service engagements supports specification of Org templates and provides design patterns that specify roles and policies. Our policy model is comparable to the $PolicyDomain$ concept of DEN-ng, but defines policies for manipulating commitments that model service engagements.

CIM Policy Model. The proposed work is intended to fit into current industry approaches, specifically DMTF’s Common Information Model [CIM, 2005]. CIM provides a generic framework for policies, including specific terms only for low-level (e.g., network-
centric) policies and actions. By contrast, the proposed approach accommodates Orgs, but policies written in our vocabulary can in principle be understood by any CIM-compliant tool.

CIM-SPL. Agrawal et al. introduce CIM-SPL [Agrawal et al., 2007], a simple policy language for CIM, which complies with the CIM Policy Model, and is a declarative language for distributed management of IT infrastructures. Each CIM-SPL policy is written within the scope of a single CIM object called anchor object. These anchor objects decide where the policy has to be authored in the policy system. Our approach provides a set of design patterns for specifying policy-based service engagements, and uses policy points to indicate where the policies have to be authored in each pattern.

Open Distributed Processing: RMODP. RM-ODP is a reference model for open distributed processing [ISO/IEC 10746 and ITU-T Rec. X.901 - X.904, 1995]. RM-ODP models several independent concerns or viewpoints, of which the enterprise viewpoint is relevant here. The enterprise viewpoint specifies a system and its environment as a community of objects (passive or active) formed for a purpose. For example, we may have a financial community consisting of people, banks, tellers, and such. An RM-ODP community resembles an Org in this respect, but it also admits purely virtual entities such as bank accounts, and even money. The enterprise viewpoint defines the Org members, their roles, and their policies. The proposed approach can be captured within RM-ODP that elaborates the enterprise viewpoint.

5.4.2 Policy Languages

Of the several policy specification languages, the following two are important. The proposed approach considers policies written as rules in Jess, and provides a policy-based architecture that is consistent with different kinds of rules, both domain specific and independent.

Rei [Kagal et al., 2003] supports deontic logic based policies by defining constructs such as rights, prohibitions, obligations, and so on. It is implemented in Prolog for pervasive environments and uses Web Ontology Language (OWL) to represent entities, access types, policy objects, speech acts, policies, and metapolicies. The architecture underlying Rei presupposes the PDP-PEP representation as introduced earlier in Section 4.1. A resource manager functions like a PEP. It consults the Rei policy engine (functioning like a PDP),
which produces a certificate specifying permissions, their validity period, and such.

Ponder [Damianou et al., 2001] is a declarative, strongly-typed, and object-oriented policy specification language for the management and security of distributed systems. It supports several policy types as event triggered condition-action rules. The Ponder policies support access control mechanisms by providing authorization, delegation, information filtering, and refrain policies. Ponder supports specification of policy constraints that are either single policy constraints, or meta-policies that express constraints on groups of policies. Ponder supports obligation policies that are event triggered condition-action rules to specify management actions. The policy syntax also supports the specification of concepts such as groups, roles, roles hierarchy, relationships and so on.

Our approach offers a high level vocabulary based on commitments and provides various design patterns capturing distinct elements of service engagements. The policies written using our vocabulary could be realized via Rei or Ponder if desired.

5.4.3 Policy Architectures

Grid research has considered Orgs and policy architectures. However, Grid policies focus on resource usage, access control, membership, and resource management.

**KAoS.** KAoS is a collection of agent services that represent and reason about policies on a variety of platforms [Uszok et al., 2003]. KAoS domain services enables agent-agent collaboration and external policy administration by capturing groups of agents, people, and other resources as organizations of domains and subdomains. KAoS policy services provides the specification and management of policies within these domains, including conflict resolution and enforcement. The domain manager ensures policy consistency and distributes them to components called *guards*, which interpret them like PDPs do. The enforcers take policies from the guards and enforce them in a platform-specific manner like PEPs do.

**Utilization Management, Accountability and Security.** Wasson et al.’s policy framework focuses on utilization management, accountability and security while specifying and enforcing policies [Wasson and Humphrey, 2003]. Their prototype policy-based VO consists of three main components: GateKeepers (representing the access points for resources), Enforcers (who carry out Org enforcement actions), and a Bank (collects resource utilization data and has information about members and resources). For every new user,
an Enforcer is created. The Bank directs the Enforcers to ensure certain actions depending on the utilization data provided by the GateKeeper. Our approach emphasizes decentralized policy-based governance as opposed to their centralized policy enforcement of VO-wide policies on resource utilization.

**Internet Communities.** A community architecture is realized with members having similar objectives, and similar resources to be shared. Feeney et al.’s architecture supports delegation of authority across communities to manage distributed resources [Feeney et al., 2004]. They support a nested community architecture with hierarchical policy enforcement, especially with respect to conflict resolution. New policies are checked for conflicts and recursively propagated to parent communities until they reach the owning community of the target resource, where they are deployed. Policy decisions at a community are made via consensus among its members. Communities are an interesting abstraction that is well-motivated by emerging usage scenarios of Orgs. A community can be supported in the proposed approach by modeling it as an Org whose policies reflect the consensus of its children. The Org agent provides a locus for the compliance with and enforcement of the community policies as well as for the conflict resolution among community members.

### 5.4.4 Policy Architectures for Trust and Privacy Management

Several policy frameworks have been developed for trust, security, and privacy management of distributed systems. A few important policy frameworks are discussed below.

**Trust management and negotiation** PeerTrust has an expressive policy and trust negotiation language based on first order Horn rules which form the basis for logic programs [Nejdl et al., 2004]. PeerTrust establishes trust between strangers with the help of dynamic exchange of certificates during an iterative privacy preserving trust negotiation process which is expressed and implemented using distributed logic programs. Role-based trust management languages such as RT [Li and Mitchell, 2003], and Cassandra [Becker and Sewell, 2004] emphasize the properties of roles such as their hierarchy. They specify role delegation, and support credential chain discovery and trust negotiation. Trust management and negotiation is relevant to the proposed approach. Orgs and other agents participating in service engagements can employ the above approaches along with the
commitments-based interactions to handle negotiation and trust management.

**Privacy preserving systems.** Several trust negotiation systems have introduced mechanisms to safeguard the privacy of the entities and their policies involved in a negotiation by using privacy preserving policies. PeerTrust [Nejdl et al., 2004] uses a protection scheme that uses named policies, so that policies can have their own policies. This way the policy written for another policy will have to be satisfied so that the policy can be revealed. PeerTrust has notions of Authority and Context and allow signed rules to facilitate privacy preserving policies. IBM’s EPAL [Ashley et al., 2003] is an enterprise privacy authorization language and is a formal language to specify fine-grained enterprise privacy policies. It provides an interoperability language for representing and exchanging data handling policies in a structured format between enterprises to enable organizations to be compliant with their stated policies, reduce overhead of configuration, and to leverage existing standards. An EPAL policy specifies lists of hierarchies of data-categories, user-categories, and purposes, and sets of actions, obligations and conditions. These elements are used in rules providing fine-grained positive and negative authorization rights. The proposed approach involves agents creating commitments toward each other, communicating and sharing information while being engaged in contractual relationships. We have not currently considered the privacy aspects in the proposed model and architecture, but our policy-based architecture can employ the above mechanisms to handle privacy.

### 5.5 Web Services Management Standards

Web Services have simplified service transactions over the Web and this area of study has resulted in several technologies to effectively manage Web Services. WS-Policy and WS-Agreement are two relevant technologies that address policy based management of web services and the formation of online agreements among the different entities such as service providers, service consumers, service brokers and so on. The Grid community has applied Web services standards to implement grid systems using Web services. Service grids architectures have applied concepts from Web services [Foster et al., 2002]. Hence there is a connection to our proposed approach which presents virtual organizations. Our proposed approach can apply some of these Web services management standards to model and manage Orgs implemented using Web services.
5.5.1 WS-Policy

Web Services Policy (WS-Policy) framework provides a policy model and syntax to describe and communicate the policies of a Web service [WS-Policy, 2006]. The model expresses the properties of capabilities, requirements, and general characteristics of entities in a XML Web services based system as policies. Policies are a collection of assertions that specify the above properties. WS-Policy provides a single consistent policy grammar for all kinds of assertions. However, it does not specify how the policies are discovered or attached to a Web service, thereby providing flexibility for other specifications.

5.5.2 WS-Agreement

The WS-Agreement specification draft, which is being developed in the GGF defines a standard way of establishing agreements between a service provider and a service customer [Andrieux et al., 2005]. The main objective here is to define a language and a protocol for advertising the capabilities of service providers and creating agreements, and for monitoring agreement compliance at runtime. This provides support for monitoring guarantees on service quality and other QoS parameters at runtime, and any failures should be notified to service consumers. Hence the agreements include state-dependent guarantees requested by the service consumers. WS-Agreement specifies an agreement establishment protocol, an XML representation of agreements and agreement templates, and a runtime agreement monitoring interface. Ludwig et al. present Cremona, an architecture and library for creation and monitoring of WS-Agreements [Ludwig et al., 2004]. The Cremona Java library provides interfaces, management functionality for agreement templates and instances, and defines abstractions for service providers which can be implemented in any domain specific environment.

The proposed approach has a proactive policy management architecture with an emphasis on monitoring and compliance checking of policies. The proposed approach can apply WS-Agreement standards to specify contracts among agents which involves many commitments that are dynamically manipulated.
Chapter 6

Conclusion and Future Work

This research focuses on modeling service engagements created among autonomous parties coming together in a contractual arrangement to share resources or carry out tasks for one another. We model enactment of service engagements in an interactive manner and at a high level. This work formalizes the atoms of a service engagement as commitments among the participants, to be created and manipulated as the engagement progresses. Further, it scopes the commitments of an engagement in an Org, and specifies how the policies of the participants affect their interactions. We propose a commitment-based architecture for Orgs. This architecture treats Orgs as consisting of agents, potentially Orgs in their own right. The nesting structure of the Orgs highlights the freedoms and constraints on the Orgs at each level. The key advantages of the proposed approach are described below:

Relationships. The proposed architecture naturally supports complex structures. It models hierarchical Orgs and partially overlapping Orgs as in exogenous Orgs. Autonomous parties from multiple administrative domains can come together and create new relationships with each other to create and enact their service engagements.

Org Modeling. The proposed approach provides a “general-purpose” and “domain-independent” distributed implementation of our architecture that supports modeling of Orgs formed in service engagements of any domain. The approach is general-purpose because we provide a generic solution to model service engagements and their enactment. The Orgs are modeled as arising within specialized Orgs called
institutions that provide the necessary social environment for the engagements formed among the members.

**Contract Enactment.** The proposed architecture provides a commitment-based approach for contract enactment. Our hierarchical and distributed Org setup supports handling of various scenarios of conflicts and exceptions in contract enactment.

**Reusable Design Patterns.** We contribute design patterns for service engagements formulated in terms of roles, commitments, and allied concepts. Each pattern reflects a distinct element of a service engagement from a business perspective and highlights exactly where policies apply. This enables the perspicuous, reusable specification of service engagements. We develop well-formedness criteria on how the patterns may be combined into engagements.

**Policy Modeling.** The proposed architecture enables us to go beyond the traditional emphasis on reactive policies. A business must not only react to explicit requests (reactive) but also must monitor its environment, collate events, and determine how to act (proactive).

**Policy-Based Governance.** The proposed architecture recognizes that Orgs are distributed. It supports two complementary perspectives. One is that there is a single locus of policy enforcement. The other is that a distributed organization must have parts that collaborate to enforce a given policy, which is achieved by a hierarchic policy aggregation as you go down the Org structure.

We have implemented a prototype called MAVOS, a policy-based multiagent system to model Orgs that demonstrates and evaluates our approach by realizing scenarios of service engagements. We considered a set of use cases in the domain of grid computing that were studied by researchers developing open grid services architecture. By considering these independently developed use cases, we showed how our approach provides a high-level modeling of service engagements and their enactment.

This dissertation opens up several directions for future research. The proposed approach opens up important directions in the field of distributed policies and service computing. The following presents a summary of the key topics which are deferred to future
work.

**Overlapping Orgs.** The proposed approach emphasizes hierarchical Orgs. But an agent can simultaneously be involved in multiple service contracts, and hence participate in multiple Orgs. Such a case suggests overlapping Orgs. The dynamics of enactment of commitments depend on the overlap and structure of the Orgs. The service engagements created in such overlapping Orgs may potentially conflict. An important challenge is analyzing the dynamics of enactment of service engagements in overlapping Orgs with potential conflicts among them.

**Additional design patterns.** This research provides a set of organizational design patterns and describes a methodology for applying them to model certain key scenarios of engagement. However, this set is far from complete, nor do we believe that there is a small complete set. Identifying additional patterns for engagement scenarios, especially handling more subtle conflicting scenarios, is a particularly important challenge.

**Handling scenarios of conflicts and dependencies among service engagements.** Our policy-based governance approach presents scenarios of preemptive scheduling applicable in service engagements and describes how we support handling of escalations. Further analysis of such scenarios to be able to automatically detect and handle exceptions leads to important directions applicable in autonomic computing.

**Studying applications of policy-based governance.** Our approach of policy-based governance is highly applicable in the area of services-based collaboration that involves content-based resource sharing and collaboration. Service engagements resulting in such areas need to respect each participant’s privacy. Extending our approach to handle additional aspects of trust and security in such collaborations is another important task.
Bibliography


Erich Gamma, Richard Helm, Ralph Johnson, and John Vlissides. Design Patterns: Elements of Reusable Object-Oriented Software. Professional Computing Series. Addison-Wesley, Reading, MA, 1995.


