Design Patterns for Policy-Based Service Engagements

Yathiraj B. Udupi
Department of Computer Science
North Carolina State University
Raleigh, NC 27695-8206, USA
ybudupi@ncsu.edu

Munindar P. Singh
Department of Computer Science
North Carolina State University
Raleigh, NC 27695-8206, USA
singh@ncsu.edu

January 20, 2008

Abstract

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.

In break from the above, recent work shows how to model 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 a (virtual) organization, and specifies how the policies of the participants affect their interactions.

This paper contributes 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. This paper develops well-formedness criteria on how the patterns may be combined into engagements.

1 Introduction

The rising prominence of services poses new challenges to computing. Services have a natural match with policies, yet existing literature on policies has not specifically addressed the challenges of services.

Service engagements in business or scientific computing 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.

Let us consider an example of how even a simple service engagement can turn out to require complex behaviors. A production grid facility creates a service engagement to provide grid services

to hurricane modelers. 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 continuously 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 [5]. The reason it is challenging is that, because it involves more than one autonomous stakeholder, 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 rely on hard coded solutions or human intervention.

Our recent work provides an agent-based conceptual model [7] and a policy-based governance architecture [8] for addressing the above kinds of challenges. It models service engagements in an interactive manner and at a high level. *Commitments* among participants are the atoms of service engagements. They are created and manipulated as a service engagement progresses. The commitments of an engagement are scoped within a (virtual) organization, termed an *Org*. The policies of the participants affect their interactions. Orgs apply in modeling service engagements by providing an architectural home for agreements among the participants. The policy-based governance approach produces proactive behaviors based on a specification of the Org of the given engagement. However, specifying an engagement directly via commitments is unwieldy. Fortunately, engagements exhibit several key repeating patterns.

This paper 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. This paper develops well-formedness criteria on how the patterns may be combined into engagements.

This paper 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* identified where each stakeholder can place its policies for carrying out desired interactions. The approach naturally support 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 paper contributes to services research a set of general-purpose computational abstractions and primitives by which service engagements can be defined.

2 Background

Consider a (fictitious, but realistic) scenario where two organizations, the National Oceanic and Atmospheric Administration (NOAA) and the National Labs Organization (NL) are involved in a service engagement wherein NL provides computing facilities to NOAA for hurricane modeling. Figure 1 illustrates this scenario (ignore the oval boxes for now). The double arrows indicate the service agreements. The dashed edges indicate the organizational structure. 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.

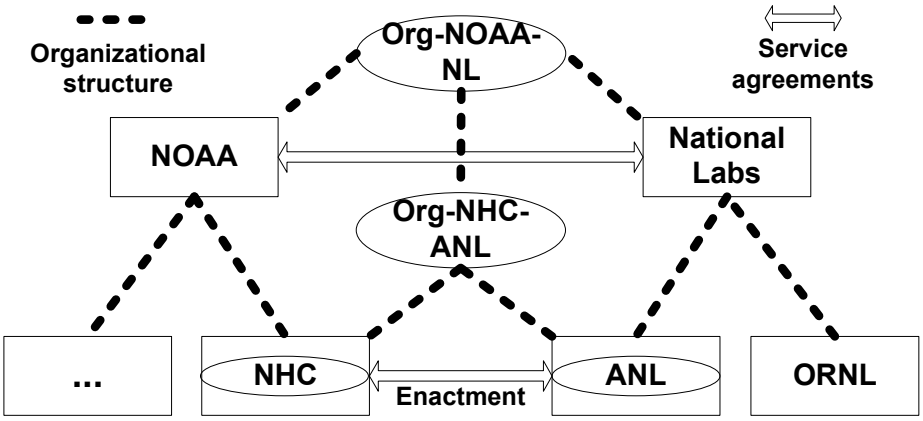

Figure 1: An example service engagement scenario

A challenging use case here is that of *preemptive scheduling* of resources in the light of critical events [5]. Consider a scenario when ANL has a service engagement with NHC as well as another organization (say, a Data Mining (DM) organization). 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, if NHC makes a request to ANL for additional resources, ANL might deny the request because of the other service engagement. This situation demands for *preemptive scheduling*, where ANL should be preempted from its lower priority service engagement with DM, so that it can satisfy the high priority request from NHC.

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. Our approach is centered on the notion of *agents*: computational entities that model autonomous, heterogeneous, and dynamic entities [7]. Orgs are multiagent systems, each individual and Org being modeled (recursively) as an agent. An Org in this sense is either a real-life organization with existing members, or is dynamically created to govern a service engagement between two or more peer agents. Agents participating in a service engagement exhibit rich policies to govern their interactions. Previous work shows how to enact such policies [8]. A service engagement is captured as a set of *commitments* between the participating agents.

Commitments. A commitment $\mathsf{C}(A, B, F, U)$ is a directed obligation from A (a *debtor*) to B (a *creditor*) to accomplish F (a *discharge condition*), but arising within the scope of U, a *context* Org. The context Org provides a means to handle exceptions by revoking or otherwise manipulating the commitment. The enactment of a service engagement proceeds by the manipulation of the corresponding commitments. Commitments may be manipulated using operations such as *create, discharge, cancel, release, delegate, assign,* and *escalate*.

For example, we may model NHC and ANL are individual agents, the other nodes are Orgs. Org-NOAA-NL is the Org for the service engagement between NOAA and NL. New Orgs are created as contexts for new relationships. For example, Org-NHC-ANL is created for the engagement between NHC and ANL. Here, ANL commits to NHC to provide the grid computing service and NHC commits to ANL to ensure accurate scheduling and supervision of the grid jobs, and to make timely payments for the services obtained. These commitments are a result of the commitments between NOAA and NL being delegated and assigned. 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. These decisions are formulated in the agents' policies.

Policies. An agent's policies govern its actions. *Domain* actions are "functional" or application operations such as running a simulation, allocating a light path in an optical network, and so on. *Communicative* actions include manipulations of commitments and administrative operations such as *request*, *accept*, *deny*, and so on. Each agent monitors and records events, which correspond to environmental observations or the actions of this and other agents. Based on the observed events and its policies, an agent may choose its actions.

In our example, ANL or NHC may send an escalate to Org-NHC-ANL. If NHC escalates, Org-NHC-ANL will apply its *escalate* handling policy before deciding its action. If the request is legitimate and if ANL cannot grant it, then Org-NHC-ANL can forward the escalate to Org-NOAA-NL. In well-designed organizations, exceptional situations are mostly handled locally, and forwarding of escalations is rare. 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 preempt ANL to give up its service engagement with DM to support NHC fully instead.

3 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 [3]. 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 2 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 resulting commitments when the corresponding decision is made.

Table 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.

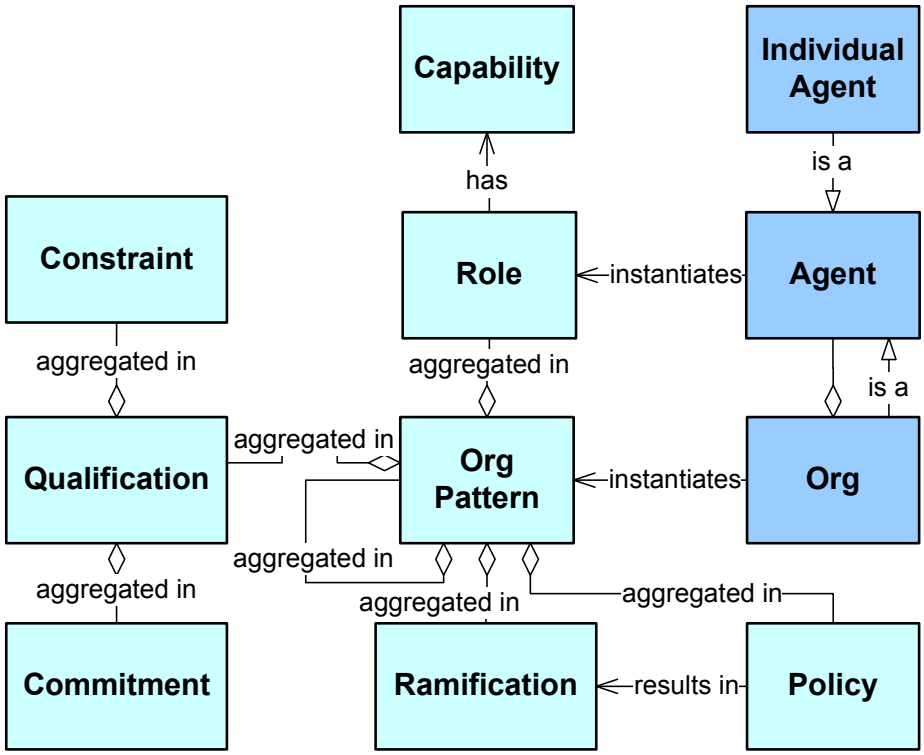

Figure 2: Org pattern model

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

Table 1: Org pattern specification schema

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} : $\mathsf{C}(D, Cr, \overline{DCr}, \phi)$: the commitment to be delegated exists $\mathsf{C}_{D'D}$: $\mathsf{C}(D', D, \overline{DD'}, \phi')$: D' is committed to D to accept (ϕ') the delegations of commitments that require ϕ **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(a), D sends a *delegate request* to D', D' *accepts* the delegate and *creates* a new commitment $\mathsf{C}_{D'Cr}$ within a newly created context Org $\overline{D'Cr}$
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*: $\mathsf{C}_{D'Cr}$: $\mathsf{C}(D', Cr, \overline{D'Cr}, \phi)$ is created **P3** (*Cr*' policy): *Cr accepts* the newly created commitment. Otherwise, the new commitment is *released*
Known uses: A merchant delegates shipping a product to a shipper

3.1 Example Org Patterns

The patterns below use the following notations. 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 ϕ and a context Org role $\overline{DCr}$. Cap_X is the set of capabilities of role X. $\phi \in Cap_X$ means that role X has the required capability to achieve ϕ. Res_{DCr} is the set of resources relevant to the commitment C_{DCr}.

An example of delegation is 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 delegater, the delegatee, and the creditor of the commitment being delegated. Figure 3(a) describes the *delegation* pattern. Here, the circles are the roles; dotted edges with solid arrows indicate existing commitments. The solid edges with arrows indicate the pattern interaction 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 and 4 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.

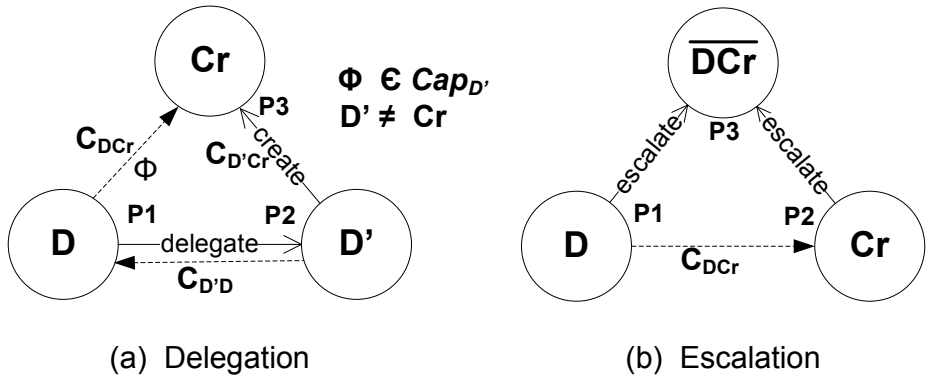

Figure 3: Delegate and escalate patterns

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.

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: $\mathsf{C}_{DCr} : \mathsf{C}(D, Cr, \overline{DCr}, \phi)$ Either of the following should occur: **Cn1**: D *cancels* C_{DCr} **Cn2**: *Cr releases* D from C_{DCr}
Scenarios: In Figure 3(b), D cancels the commitment. D or *Cr* may send an escalate to $\overline{DCr}$. $\overline{DCr}$ 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 $\overline{DCr}$ **P2** (*Cr*'s policy point): If *Cr releases* D from C_{DCr}, it *escalates* it to $\overline{DCr}$ **P3:** ($\overline{DCr}$'s policy point): $\overline{DCr}$ *accepts* escalates from D or *Cr* $\overline{DCr}$'s escalate handling policy would determine if the escalate is forwarded up the Org hierarchy, or if $\overline{DCr}$ locally handles the escalated commitment. $\mathsf{C}_M : \mathsf{C}(Dm, Cr, \overline{DmCr}, \phi)$, is the resulting commitment after resolution, where *Dm* is a compensating debtor
Known uses: A customer complains if an ordered shipment does not arrive

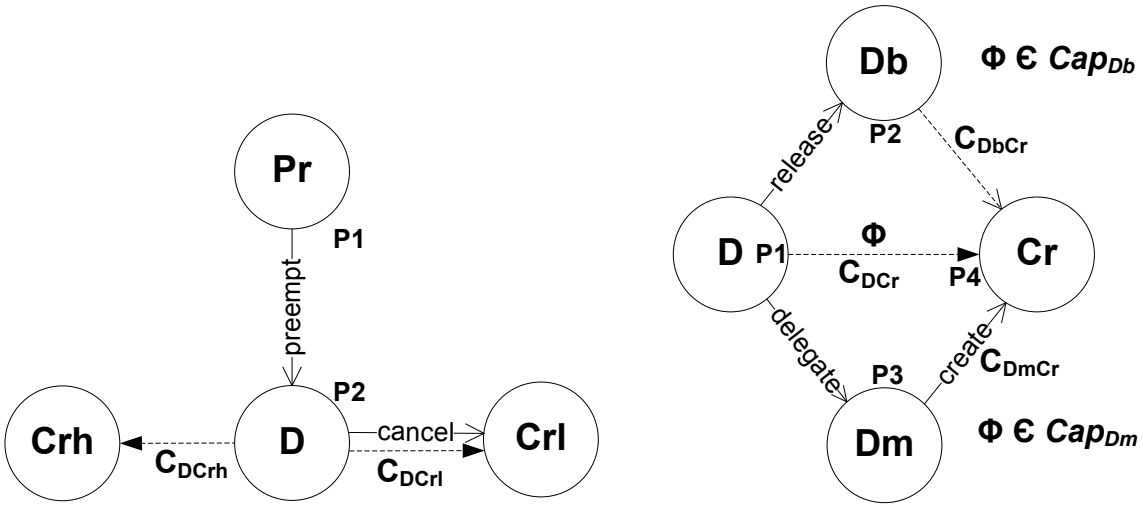

Figure 4: Preemption and Reallocation patterns

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 2 demonstrated *preemptive scheduling* as a way to resolve conflicting situations. The next two patterns capture these scenarios.

Name: ***Preemption*** (D, Crl, Crh, Pr) Here, *Crl* is a lower priority creditor than *Crh*, and *Pr* is the preempting role
Purpose and applicability: To cancel a commitment based on conflicting demands
Qualifications: C_{DCrl} : $\mathsf{C}(D, Crl, \overline{DCrl}, \phi_1)$: the lower priority commitment of D exists C_{DCrh} : $\mathsf{C}(D, Crh, \overline{DCrh}, \phi_2)$: the higher priority commitment of D exists **Cn1:** *Pr* has set a higher priority for C_{DCrh} than C_{DCrl}
Scenarios: In Figure 4(a), D is unable to discharge both the commitments C_{DCrl} and C_{Dchl} 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 delegater, 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

For brevity we do not show other patterns in detail. Figure 4(b) illustrates the pattern ***Reallocation***$((D, Cr, Db, Dm)$. Here, a delegatee (*Db*) that cancels a delegated commitment is released and the original commitment is redelegated to another agent (*Dm*). Here, a cancel is implied on timeouts (when a violation occurs). This applies when an agent is involved in multiple contracts simultaneously. For example, 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. The pattern *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. *Assign*(D, *Cr*, *Cr'*) corresponds to a commitment C_{DCr} being assigned by *Cr* to *Cr'*. Similarly design patterns can be described for the commitment manipulation scenarios such as *create*, *cancel*, *discharge*, and *release*. More sophisticated patterns include *Collaboration*, *Backup*, *Separation of Duties*, and *Least Privileges*.

3.2 An Example Service Engagement

A service engagement is specified using the above patterns. Each engagement specification follows the template of Table 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.

The following formalize the NOAA–NL service engagement using the above approach.

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 C_3: C(*SP*, *SC*, *Org-SP-SC*, ϕ), where ϕ means the compute service is provisioned successfully. Commitment C_1 is first created that captures the engagement between SPO and SCO within Org Org-SPO-SCO. SCO assigns C_1 to SC to create an assigned commitment C_2. C_2 is now delegated by SPO to SP to create C_3.

Qualifications: These include the delegation, assign, escalation, and the preemption patterns as illustrated in Figure 5 and described below:

Assign (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 commit-

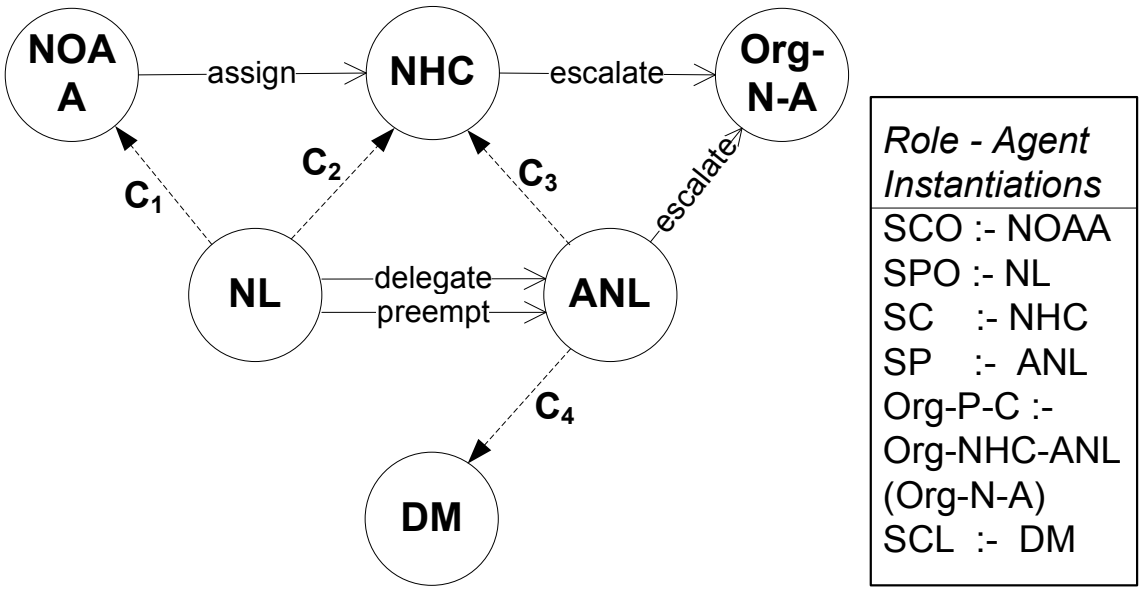

Figure 5: Service engagement instantiations

ment 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 preempting Org that preempts *SP* from the lower priority commitment.

4 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.

4.1 Structure Based on Design Patterns

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

Definition 1 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 *delegater* D can delegate a commitment to D', a *delegatee*.

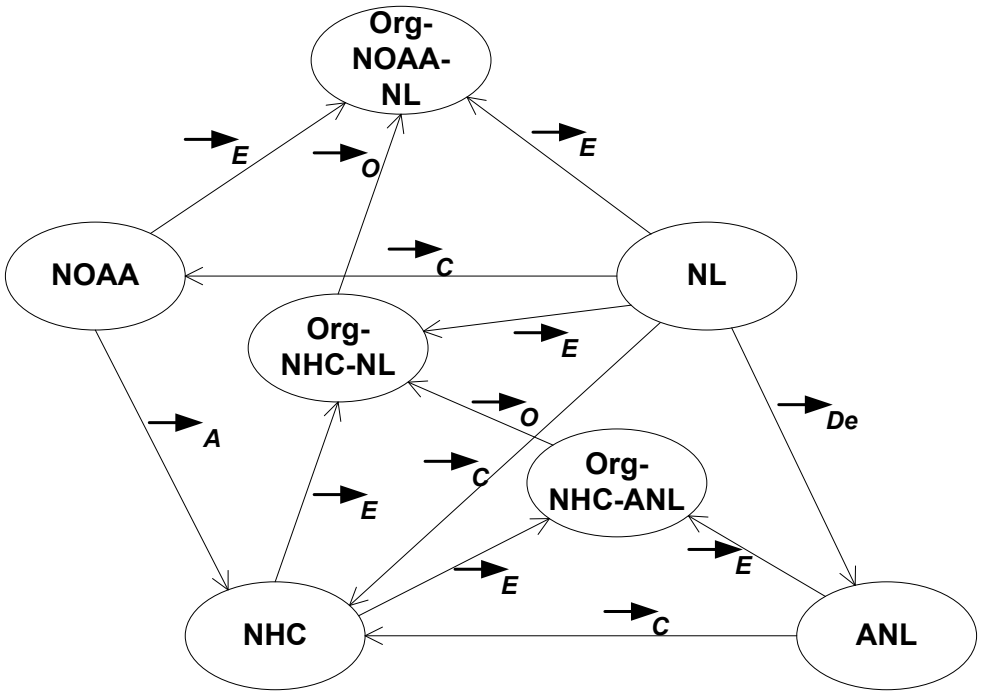

Figure 6: Org structure with role links

- ***Assignment link*** $\langle$*Cr* $\rightarrow_A$ *Cr'*$\rangle$: A creditor *Cr* can assign a commitment to an *assignee Cr'*.
- ***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*** $\langle R \rightarrow_E \overline{DCr}\rangle$: A role R, which could be either D or *Cr*, can escalate C_{DCr} to the context $\overline{DCr}$.
- ***Org Escalation link*** $\langle O_c \rightarrow_O O_p\rangle$: Org O_c is a child of O_p, and can forward an escalate to O_p.
- ***Release link*** $\langle Rr \rightarrow_R D\rangle$: A role Rr (creditor or context) can release a debtor D from its commitment.

Table 2 shows the roles and role links introduced by Org patterns. Figure 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 3.2.

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 *Rcl*.

Pattern	Roles	Role links
Delegation	D, Cr, D'	$\langle D \rightarrow_C Cr\rangle$, $\langle D \rightarrow_{De} D'\rangle$, $\langle D' \rightarrow_C Cr\rangle$
Assign	D, Cr, Cr'	$\langle D \rightarrow_C Cr\rangle$, $\langle Cr \rightarrow_A Cr'\rangle$, $\langle D \rightarrow_C Cr'\rangle$
Escalation	$D, Cr, \overline{DCr}$	$\langle D \rightarrow_C Cr\rangle$, $\langle D \rightarrow_E \overline{DCr}\rangle$, $\langle Cr \rightarrow_E \overline{DCr}\rangle$
Create, Cancel, Discharge	D, Cr	$\langle D \rightarrow_C Cr\rangle$,
Release	D, Cr, Rr	$\langle D \rightarrow_C Cr\rangle$, $\langle Rr \rightarrow_R D\rangle$

Table 2: Representative pattern roles and role links

Preemption$(D, Crl, Crh, Pr) =$ ***Create***$(D, Crl) \oplus$
Create$(D, Crh) \oplus$ ***Release***(D, Crl, Pr)
Role-links $= \{\langle D \rightarrow_C Crl\rangle, \langle D \rightarrow_C Chl\rangle, \langle Pr \rightarrow_R D\rangle\}$

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*.

Reallocation = Release$(Db, Cr, D)\oplus$
Delegate$(D, Cr, Db) \oplus$ ***Delegate***(D, Cr, Dm)
Role-links $= \{\langle D \rightarrow_C Cr\rangle, \langle D \rightarrow_{De} Db\rangle,$
$\langle Db \rightarrow_C Cr\rangle, \langle D \rightarrow_{De} Dm\rangle,$
$\langle Dm \rightarrow_C Cr\rangle, \langle D \rightarrow_R Db\rangle\}$

4.2 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 assign pattern instantiations should be accompanied by escalate 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 $\langle R_3 \rightarrow_E$ Org-R_3-$R_2\rangle$, $\langle R_2 \rightarrow_E$ Org-R_3-$R_2\rangle$, $\langle$Org-R_3-$R_2 \rightarrow_O$ Org-R_1-$R_2\rangle$. In Figure 6, ***Delegation***(NL, NHC, ANL) is accompanied by ***Escalation***(ANL, NHC, Org-NHC-ANL) and ***Assign***(NL, NOAA, NHC) is accompanied by ***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 $\langle R_1 \rightarrow_C R_2\rangle$ exists, then a sound service engagement should have $\langle R_1 \rightarrow_E$ Org-R_1-$R_2\rangle$ and $\langle R_2 \rightarrow_E$ Org-R_1-$R_2\rangle$, i.e., a commitment creation should be accompanied by an escalate pattern instantiation.

In Figure 6, ***Create***(NL, NOAA) is accompanied by ***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 non-terminating service engagements, and hence required for *completeness* of a service engagement.

Postulate 3 If $\langle R_1 \rightarrow_{De} R_3 \rangle$ ($\langle R_1 \rightarrow_{A} R_3 \rangle$) exists, then a *complete* service engagement should not have $\langle R_3 \rightarrow_{De} R_1 \rangle$ ($\langle R_3 \rightarrow_{A} R_1 \rangle$).

For example, in Figure 6 $\langle$NL $\rightarrow_{De}$ ANL$\rangle$ and $\langle$ANL $\rightarrow_{De}$ NL$\rangle$ cannot both exist. If both exist, then it may lead to non-terminating service engagements because of cyclic delegations.

In a service engagement specification, if multiple delegations from different delegaters 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 $\langle Pr \rightarrow_{R} D' \rangle$, where *Pr* is the preempting role.

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.

5 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.

We have implemented a prototype using a policy engine based on Jess and messaging middleware that demonstrates the policy-based enactment, including the above scenarios.

Future work. This paper opens up important future directions in the field of distributed policies and service computing. Specifying design patterns for handling more subtle conflicts among service agreements is left to future work.

5.1 Related Work

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, and motivating design patterns to specify such engagements.

Policy Models and Languages 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 [6]. DEN-ng defines concepts such as *PolicyConcept*, 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 for design patterns includes roles and policies, and is comparable to the *PolicyDomain* concept of DEN-ng, but defines policies for manipulating commitments that model service engagements.

Agrawal *et al.* introduce CIM-SPL [1], 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. Rei [4] supports deontic logic based policies by defining constructs such as rights, prohibitions, obligations, and so on. Ponder [2] supports several policy types as event triggered condition-action rules. Our approach offers a high level vocabulary based on commitments and provides various design patterns capturing distinct elements of service engagements. It could be realized via Rei or Ponder if desired.

Roles, Agents, Organizations Wooldridge *et al.* present *Gaia* as a methodology that provides an organizational understanding of agent systems [9]. An organization is viewed as a collection of roles that relate to other roles and take part in interactions. 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.

References

[1] Dakshi Agrawal, Seraphin B. Calo, Kang-Won Lee, and Jorge Lobo. Issues in designing a policy language for distributed management of it infrastructures. In *Integrated Network Man-*

agement, pages 30–39, 2007.

[2] Nicodemos Damianou, Naranker Dulay, Emil Lupu, and Morris Sloman. The ponder policy specification language. In *Proceedings of International IEEE Workshop of Policies for Distributed Systems and Networks (POLICY)*, pages 18–38, 2001.

[3] 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.

[4] Lalana Kagal, Tim Finin, and Anupam Joshi. A policy language for a pervasive computing environment. In *Proceedings of 4th International IEEE Workshop on Policies for Distributed Systems and Networks (POLICY)*, pages 63–74, June 2003.

[5] NGG2. Next generation grids 2: Requirements and options for european grids research 2005–2010 and beyond, July 2004. Expert Group Report at www.semanticgrid.org/docs/ngg2_eg_final.pdf.

[6] John Strassner, Jose Neuman de Souza, David Raymer, Srini Samudrala, Steven Davy, and Keara Barrett. The design of a new policy model to support ontology-driven reasoning for autonomic networking. *Network Operations and Management Symposium, LANOMS. Latin American*, pages 114–125, Sept. 2007.

[7] Yathiraj B. Udupi and Munindar P. Singh. Contract enactment in virtual organizations: A commitment-based approach. In *Proceedings of the 21st National Conference on Artificial Intelligence (AAAI)*, pages 722–727, Menlo Park, July 2006. AAAI Press.

[8] Yathiraj B. Udupi and Munindar P. Singh. Governance of cross-organizational service agreements: A policy-based approach. In *Proceedings of the IEEE International Conference on Services Computing (SCC)*, pages 36–43, July 2007.

[9] Michael Wooldridge, Nicholas R. Jennings, and David Kinny. The Gaia methodology for agent-oriented analysis and design. *Autonomous Agents and Multi-Agent Systems*, 3(3):285–312, 2000.