Multi-Domain IT Architectures for Next Generation Communications Service Providers

Rob Brennan\(^{1}\), Kevin Feeney, John Keeney, Declan O'Sullivan
Trinity College Dublin \{rob.brennan, kevin.feeney, john.keeney, declan.osullivan\}@cs.tcd.ie

Joel J. Fleck II
Hewlett-Packard, Office of Strategy and Technology, \{joel.fleck@hp.com\}

Simon Foley
University College Cork \{s.foley@cs.ucc.ie\}

Sven van der Meer
Waterford Institute of Technology \{vdmeer@tssg.org\}

1 Challenges for Next Generation Service Providers

The boom in Internet-based services has challenged traditional Communication Service Providers (CSPs) as they strive to avoid becoming purveyors of commodity bit-pipes. CSPs are seeking to leverage their customer knowledge, enterprise-quality services, billing and security infrastructures to widen their customer and revenue base. The agility and adaptability of CSPs’ Service Delivery Platforms (SDPs), in terms of their support for new business models, ability to selectively open the network and engage in value networks with customers and suppliers are widely seen as keys to CSPs achieving their full business potential.

This trend, along with information and communications technology convergence, has led to a renewed interest in the application of proven IT technologies to CSP SDPs and Operations, Administration and Maintenance (OAM) infrastructures. IT technologies present two challenges for CSPs: to support the traditional stringent real-time and scalability requirements and to provide the required flexibility needed for distributed multi-vendor environments. For example, one popular IT best practice service governance model is the IT Infrastructure Library (ITIL) [1]. ITIL version 3 offers a comprehensive approach to governing the creation, design, development, deployment, operation, change management, and eventual termination of services. It has also been considerably re-worked to directly reference Service-Oriented Architecture (SOA) concepts, whereby combining the strong governance model in ITIL with the flexibility of SOA, CSPs can build adaptable, value-driven, re-usable systems. However, how can this process-centric approach be applicable to the fractured business models and proliferation of multi-enterprise value chains based on virtual operators, software as a service and cloud-based solutions in the CSP space? Dictatorial, end-to-end, process-based management systems that require unfettered control and reliable global knowledge are of limited use in the dynamic multi-organisational environments that are becoming the norm for CSPs. Similarly, modern trends in OAM have stressed the

\(^{1}\) Corresponding author
deployment of autonomic and adaptive systems in order to reduce operational expense (OPEX) and increase flexibility, but current work has largely focused on centralized architectures and single-CSP control systems.

Ignoring inter-domain systems management and the organizational complexity of modern enterprises means that no matter how adaptive or autonomic an application (and the network it runs on) is, it is still basically a static application unless the network sensing, analysis, and control systems can adapt with the multi-domain managed system as its solutions and applications reconfigure. To do this without explicitly and dynamically managing the relationships between the domains contributing to the communications service delivery is ineffective. In addition, the inherent and incurable heterogeneity of inter-domain systems means that the network and service descriptions used for management must be extended beyond the static semantic description of their components. Thus, we identify multi-domain relationship management based on semantic (knowledge) models as crucial to the deployment of IT infrastructure supporting next-generation networks in CSPs.

2 Conceptual Model for Relationship Management

Having identified the importance of managing relationships and organizational diversity for the ultimate success of flexible IT systems supporting modern CSPs – we explore the question of how this can be achieved in this section.

2.1 Definition of Domains

Before discussing domain relationships, it is important to first define what we mean by domains. In this work a domain (or management domain) is defined as an autonomous (self-governing), administrative entity that has a specific business role and consequent internal goals and local policies [2]. It has a clear boundary defined by the scope of its authority over resources or artefacts. Example domains are business units, e.g. customer billing, within an organization or whole enterprises, for example a third-party CRM company.

However, domains do not exist in a vacuum. In order to deliver useful services and business value they often co-operate with or use the capabilities of other domains. These relationships with other domains must themselves be managed: this requires an explicit model of the relationships. Two important aspects of a relationship model are modelling shared capabilities and the specification of the operational rules that govern the use of those capabilities. These are both well-known features of traditional OAM models. Unlike traditional OAM systems, multi-domain systems lack a central authority guaranteeing interoperability and need:

- To define or negotiate both shared semantics and a set of protocols for decision-making, conflict resolution, and relationship membership
- Trust management to enable domains to communicate with an appropriate level of information security and guarantees of relationship integrity
- Relationship **auditing and monitoring** to provide evidence of long-term conformance to the agreed parameters of a relationship.

### 2.2 A Layered Relationship Model

In this work the term *domain relationship* is employed to describe cross-organizational capability sharing agreements. However, organizational arrangements between autonomous entities vary widely in scope and can be complex and multi-faceted. Thus, models of relationships must be capable of capturing and reflecting the most important factors that vary across such arrangements if they are to support modeling the evolving, dynamic nature of the real world. These relationships can be hierarchical, whereby we call them *domain compositions*, or peer-to-peer, whereby we designate them *domain federations*.

The Layered Relationship Model (cf. Figure 1), is a general-purpose conceptual model of the components of a domain relationship. The model is decomposed into layers, with each layer representing one aspect of the organizational arrangement. This layered model should not be confused with a communications stack—in some relationships there may be cross-layer interactions and layers may be empty. Its main purpose is to serve as a useful model for decomposing relationships in order to render their definition and maintenance more tractable and transparent. The layers represent the most important aspects of cross-organizational relationships for successful persistent organizational relationships, with their relative positioning in the layered model representing the dependencies between the elements that constitute such an agreement. In the next section we deal with concrete mechanisms for instantiating this model.

![Layered Relationship Model (LRM)](image)
3 A Multi-Domain Relationship Management Architecture

This section describes an architecture for multi-domain IT systems relationship management based on three key components:

- The Domain Relationship Map (DRM) - an explicit model of the relationship from an individual domain’s perspective. It supports aspects of LRM operational rules, shared capabilities, shared semantics and relationship definition layers;

- The Trusted Community Based Policy Management System (TCBPMS) – a secure, distributed model of multi-domain relationship structures and shared capability authorities (access controls or other operational rules). It supports LRM shared capabilities and distributed operational rules. It also includes an infrastructure for distributed trust management that supports the LRM trusted communication layer;

- A relationship Traceability Map tool chain – to support relationship context tracking, contract-policies analysis, and contract stub code generation.
The relationships are coordinated and managed by peering, context-aware Domain Relationship Managers (Figure 2). The Domain Relationship and Traceability Maps described below are core tools facilitating this adaptive management. The use of technology neutral models allows management systems to verify semantic consistency across the relationship using ontologic reasoning. The use of ontologies in conjunction with representations of business goals, policies and associations allows reasoning about the compatibility or consistency of proposed or existing relationships. The use of technology neutral models also facilitates development of implementations of each side of a relationship using an approach based on Model Driven Architecture (MDA) reducing handcrafting and testing.

The TCBPMS provides support for integration with the security infrastructure and enforces distributed access controls on inter-domain contracts (service interfaces) but also supports distributed, secure formation and management of the relationships between the domains. The sub-sections below describe these components and the generated artifacts in more detail.

The Domain Relationship Manager is also responsible for coordinating with its peers from other domains to decide when to mediate, accept or terminate relationships. It needs to handle both domain Composition and Federation. In the case of composition, both the external ontologies and the external Relationship Maps must be consistent. If any inconsistencies exist, the management system must either mediate and resolve them or the relationship must be abandoned. Failure to achieve consistency across the relationship would amount to domains within an organization working toward different, potentially conflicting goals.

Federations only require the ontologies of the participating domains to be consistent, while the Goals, Policies, and Associations of the DRM must be compatible. Incompatibilities between Federated Domains trigger either a negotiation phase between Relationship Managers to resolve the incompatibilities or abandoning the Federation to search for a replacement Domain satisfying the consistency and compatibility requirements.

### 3.1 The Domain Relationship Map

To dynamically create, modify and remove relationship agreements between domains, we introduce the Domain Relationship Map [3] concept, which models each side of the relationship. It describes, based on the knowledge model of the exposing domain (describing internal Business Strategy, Business Goals, Business Rules, Local Policies, and Resources), an ontology-based knowledge model for exported capabilities, the goals of the potential relationship, and the policies governing it. Not only must the Relationship’s Ontology (knowledge model) be consistent with (and a subset of) the internal Ontology of the Domain, but the Business Strategy, Goals, and Rules of the Relationship also must be consistent with those of the Domain.

These models are used in two ways:
1. To manage the lifecycle of the relationship by assuring:
   a. that the model on each side of the relationship remains consistent with the internal model of its associated domain,
   b. that both models on each side of the relationship remain compatible (or consistent – depending on what type of relationship is implemented) with the model on the other side of the relationship;

2. As the basis for a model driven approach for deriving and producing the technology specific software artefacts that implement the relationship.

The Business Goals of the Relationship are documented in SBVR (described below) in two parts: a vocabulary which is consistent with the Ontology of the Relationship; SBVR Rules which must be consistent with Business Rules of the Domain. The SBVR Vocabulary and Rules describing the Relationship are used to develop the Technology Neutral descriptions of the policies and capabilities of the relationship using model-driven transformations. The Fact Types of the SBVR Vocabulary are used as the basis for a transformation to the XPATH 2.0 assertions that describe the capabilities desired or offered by the Relationship. The SBVR Rules are used as the input for a transformation to the technology neutral N3 representation of the policies governing the operation and management of the relationship.

### 3.1.1 Key Technologies for Relationship Modeling

Many of the technologies deployed in the DRM are not well-known by non-knowledge engineers and so we present brief overviews here.

#### 3.1.1.1 Model Driven Architecture (MDA)

The Object Management Group (OMG) has defined the Model Driven Architecture (MDA) approach as a methodology for developing distributed applications by defining a Computational-independent Business Model (CIM) which is transformed to a Platform-Independent Base Model (PIM) and to one or more Platform-Specific Models (PSMs) and interface definition sets. Using this approach, neither the CIM nor the base PIM describing the application behavior need to be changed to support modifications in implementation technologies. The model driven approach described in this paper extends this approach by using transformations to provide tools to validate consistency of inter-domain relationships with the internal specifications of the participating domains, and by using transformations to generate both policies governing the relationship, the code implementing the relationship and the tools needed to monitor and maintain its health.

#### 3.1.1.2 Resource Description Framework (RDF) and Web Ontology Language (OWL)

Resource Description Framework (RDF) is the W3C standard for meta-data [4]. It is based on making three-part entity-attribute-value assertions called “triples” that can be combined into a directed graph-based data, information, or knowledge representation. The base RDF specification has no inherent
typing mechanisms until it is extended with the RDF Schema (RDFS) specification. RDF/RDFS are especially useful for flexibly representing data about entities that have a very large range of potential attributes, e.g. if the attributes are unknown, such as when they are defined across a domain boundary where alien concepts may be modelled or common concepts may be modelled in unfamiliar ways. The second property of RDF that is especially useful in cross-domain modelling is the ability to easily merge data about a single entity from multiple sources.

The Web Ontology Language (OWL) [5] is a set of semantic web representation languages designed to capture ontological concepts, instances of concepts and relationships. OWL builds extensively on RDF/RDFS, but provides additional vocabulary and semantics to better capture the meaning of concepts, relationships, and instances. OWL ontologies are commonly made available in RDF/XML format. One of the main advantages of OWL’s formal semantics over other ontological representation approaches is the ability to use automatic inference engines to extract additional semantic statements that were implicit in the ontology and make them directly accessible.

### 3.1.1.3 SBVR

Semantics of Business Vocabulary and Rules (SBVR) is a standard [6] developed by the OMG to facilitate formal documentation of the goals governing the operations of a business. Traditionally, businesses use operational guidelines defined across ad-hoc documents lacking formal structure to document business goals. The SBVR specification provides formal semantics to enable consistent understanding and interpretation of the operating principals of businesses. SBVR is independent of information systems and thus provides a means of bridging the gap that commonly exists between business operational guidelines and the systems that implement them. To facilitate the interchange of data and to provide for standardized data interfaces between entities, artifacts or tools, the SBVR meta-model generation process uses a structured, fact-oriented approach incorporating a formal business vocabulary and a set of business rules built upon that vocabulary.

An extract of the SBVR Metamodel for a distributed billing system is presented in Figure 3.
Consumer
Definition: person or resource proxy

Customer
Definition: entity the pays for goods or services
Synonym: client
General Concept: Consumer

Customer Type:
Definition: concept that specifies the Customer and that classifies a Customer based on a Service Level
Concept Type: categorisation type

Platinum Customer
Concept Type: Customer Type
Definition: Customer who is charged 5 per unit

Gold Customer
Concept Type: Customer Type
Definition: Customer who is charged 2 per unit

Silver Customer
Concept Type: Customer Type
Definition: Customer who is charged 1 per unit

Rule 8: It is obligatory that the Rater assign a Customer Type to a Normalised Usage Record
Rule 9: It is obligatory that each Customer belongs to one and only one Customer Type

Figure 3: Sample SBVR Vocabulary and Rules

3.1.1.4 N3

Notation3 (N3) [7] is a language designed to be a compact and readable alternative to RDF/XML. It is used in the DRM to model policies governing relationships in a technology neutral manner. In addition to supporting the full expressiveness of RDF, N3 also supports the encoding of RDF rules logic. An important aspect of N3 is the ability to explicitly specify hypothetical sub-graphs for use in rules or to load graph sub-sections from other sources. In the context of this work, this approach can be exploited to define operational policy rules based on the contents of the knowledge-base (see [8] for an example).

For example, the following N3 snippet (fig. 4) allows people from “ABC_Corp” working on the collaborative “FAME” project to access sales orders in “XYZ_Corp” relating to the “FAME” consortium.

```
@forAll :access_request .
{    ?access_request a xyz:DataAccessRequest .
    ?access_request.target a xyz:SalesOrder .
    ?access_request.target xyz:relatedToProject [    xyz:projectTitle "FAME"    ] .

    <ABC_Corp_Employees.n3> log.semantics emp_abc .
    emp_abc log.includes { [ ] a abc:employee ;
        abc:hasName ?access_request.requester ;
        abc:worksOnProject "FAME project" } .
}
=> {    ?access_request xyz:isAuthorised "Authorised" } .
```

Figure 4: Sample N3 Rule
This example illustrates an interoperability challenge whereby different organizations represent information differently. In “XYZ_Corp” ontologies, the “Fame” project is represented as an ontological instance, whereas in the “ABC_Corp” a project is only a string value used to tag employees.

### 3.2 Trusted Community-Based Policy Management System

The Community-based Policy Management System (CBPMS) \[9\] is a distributed policy management approach for federated systems. It utilizes a flexible, graph-based capability authority model to partition and delegate federated capabilities or services as delegation chains. The Trusted CBPMS solution extends the basic CBPMS features with trust management functions that address threats from malformed or malicious federated principals and provides increased flexibility in delegation chain reduction and local capability authority re-partitioning. Dedicated policy logic supports secure decentralized reasoning within TCBPMS with an implementation based on public-key certificates.

### 3.3 Trust Management

Network communications security technologies, such as IPsec can be used to ensure secrecy, integrity, and authenticity of messages exchanged between domains. However, domains may make fraudulent statements over these secure channels about their capabilities or the capabilities of others. This threat may be due to inadvertent or malicious intent on the part of the domain's business process, or as a consequence of an intruder compromising part of the federation-supporting infrastructure (e.g., a policy server). Therefore, TCBPMS provides not just secure channels between domains, but also secure formation and management of relationships between the domains.

In centralized architectures it is relatively straightforward to provide this security infrastructure since all related policies, capabilities, etc., are centrally stored and managed on a secure host. However, as autonomous self-governing entities, domains should not have to rely upon some central authority when forming and managing relationships. This domain self-determination is achieved by taking a Trust Management approach to distributing the Layered Relationship model (the policies, capabilities, etc.) across the domains whereby federation policy decisions can be made locally in a domain without reference to any central authority. Trust Management uses cryptographic certificates to implement this secure distribution and ensure that one domain cannot make fraudulent statements about the capabilities of another.

#### 3.3.1 Capability Authority Models

Capability authority models describe how the capabilities shared by a domain are bundled together into sets of capabilities and specific associated permissions. These bundles are known as capability authorities. A fundamental aspect of the capability authority approach lies in the ability to compare two capability authorities and decide whether the first capability authority encapsulates the second according to the model. This allows capability authorities representing arbitrary aggregations of specific permissions to be distributed between federated domains. Whenever a third-party invokes a capability, the federated domain merely establishes whether the capability being invoked is encapsulated by a
capability authority that has been issued to that domain. In addition to maintaining the capability authority graph, it is necessary for the distributed TCBPMS instances to maintain a capability authority delegation graph which indicates where a given capability authority resides.

Capability models thus provide an access control mechanism across domain relationships. Capability authorities can also be associated with policy rules defined within the management system of the domain that controls the capabilities. This provides a flexible and expressive means of applying access control to capability sharing in federal relationships without requiring that all parties support common policy or information models.

In addition to capability discovery and description, infrastructure is required to enforce security requirements – so that exposed capabilities can only be used by those parties to whom they have been shared and that this use does not inadvertently expose any information that might help hostile parties from gaining access to confidential information. Additionally, it is generally desirable to limit the amount of internal information that is exposed to third parties to the absolutely minimum necessary.

To tackle these problems, the TCBPMS provides a trusted capability authority architecture which aims to fully insulate the exposed capabilities from internal processes. The trusted capability authority layer cryptographically signs each shared capability authority when it is distributed to third parties. When a third-party attempts to use the capability, the signature is checked to ensure that the specific capability being used has been shared with that third-party. Arbitrary re-packaging of signed, shared authorities is supported as this helps with manipulation by different domains in terms of their local policy or resource models.

### 3.4 Tool Chain for Automatic Generation of Traceability Maps

Traceability Maps (TM) extend the understanding of context using a graph to document the relationships of software artefacts (Business Goals, Policies, Contracts, and Processes) throughout the Lifecycle of software [10]. The Traceability Maps for each type of software artefact can be combined to form an intra-Domain Traceability Map documenting the relationships between all software artefacts. In a manner similar to the construction of intra-Domain Traceability Maps, a TM can be constructed for each inter-Domain relationship and combined with the intra-Domain Maps resulting in a full solution TM. The automatic generation of Traceability Maps is realized using a model-driven approach and a number of Domain Specific Languages (DSLs). We use the DEN-ng information model as the foundation, specifically the policy model, combined with a contract model based on TeleManagement Forum TMF 053b. In a manual step, we define a stratified policy language and a contract language (EBNF), which in turn provides the basis to develop a tool chain to apply them for generating Traceability Maps. The policy language represents a high-level language following the classification of the DEN-ng policy model and provides dialects to address the different needs of various constituencies (i.e. business analyst and system administrator). The contract language represents a formal definition for contract-based components. Both languages allow a relationship to be defined with the contract part specifying the
agreed conditions and rules of the relationship and the policy part specifying the control of the relationship, for instance in cases where contracts are violated.

Figure 5: Traceability Map Tool Chain

Figure 5 shows the implemented tool chain that supports the automatic generation of Traceability Maps. The policy language is named DPOL and the contract language is named L-ADS. We have developed two different set of tools, one based on xTeXT (Eclipse model-based development environment) and one based on ANTLR (a parser generator). With xText, we can generate editors as Eclipse plug-ins and then handover the resulting policy and contract specifications to an ANTLR compiler which transforms between different textual representations. Finally, we can populate our (DEN-ng) knowledge base with all information needed to generate Traceability Maps (mid-right part of the figure). The knowledge base can handle intra- as well as inter-domain maps. We also have developed a set of clients (Java Policy client, Conflict Detection) to analyze the generated maps.

4 Use Case – Service Billing as a Service

In order to illustrate the advantages of our approach to building multi-domain IT systems we provide this use case for a key SDP enabler – a comprehensive billing function.

4.1 Service Structure

Figure 6 illustrates a billing function that is itself a multi-domain service that includes third-party components serving a number of CSPs to provide a unified billing solution (one of the distinguishing features of CSP networks). In a next-generation network the number of domains traversed for service
delivery increases and Flexible billing architectures make it more likely that CSPs will be able to appropriately meet the varied needs of customers. Hence it is natural to allow out-sourcing of specialised expertise in specific aspects of the billing flow. The disadvantage of this approach for a traditional IT architecture is that, as the number of relationships or integration points grows, the more likely the system is to become brittle and inflexible since each external and internal relationship must be maintained by custom gateways. In our approach, formal models of the relationships keep the integration points lightweight, adaptable and explicitly manageable because the relationship models allow automated or semi-automated negotiation and re-negotiation of the interface capabilities and the mappings between the exported or imported capabilities and internal policies and models.

Figure 6: Use Case: Billing Domain Service Structure

4.2 Dynamic Behaviour

The sub-sections below discuss the applicability of our domain relationship manager architecture to the use case.

4.2.1 Domain Matching in Action

Emerging billing environments may use different suppliers to support presentation of intermediate or final bills to the customers: 3rd party presentation vendors are selected “on-the-fly” based upon customer needs (cost, availability, performance, type of bill presentation requested), context (location, time) or many other criteria derived from the business strategy of the bill provider and the desires of the bill consumer. In such an environment, it is essential that, rather than long-term static inter-domain
bindings which tend to be a compromise for the billing provider and an “average” customer (and thus not meeting the business goals and strategy of either party), inter-domain relationships are dynamic and flexible to allow the best available fit between specific customers’ desires and the capabilities of the billing presentation service providers.

4.2.2 Capability Discovery
To negotiate sharing regimes between domains, the parties must support discovering, reasoning and negotiating about the capabilities that other parties could make available. Fully automating this discovery and negotiation is an extremely difficult problem – since the nature of negotiation is such that even revealing which capabilities are available for sharing may be valuable information that could compromise a party’s bargaining position. Therefore, we assume that the parties to such federations have pre-existing legal relationships and are already in a situation where they know which third-party capabilities that they wish to utilize. This assumption allows us to concentrate on the problem of relationship participants translating the capability authorities that they have been delegated into sets of capabilities that they can invoke. Our approach is to provide semantically rich descriptions of capability authorities, using RDF to describe the semantics of the capabilities being shared and the set of permissions that they encompass. This allows third parties who have been delegated particular capability authorities to use SPARQL-based queries to translate their capability authorities into concrete capabilities that they can directly invoke.

4.2.3 Establishing Trusted Relationships
A customer trusts billing from its CSP, which in turn federates with a third-party billing presentation service by signing a capability (certificate) that delegates authority on billing to the third-party. This certificate provides unforgeable proof to the customer that it can trust billing information from the third-party without having to interact with the CSP. This is a simple scenario: in practice, certificates encode the policies, capabilities, etc. across the Layered Relationship Model that are related to federations involving this third-party.

5 Conclusions
We have presented a comprehensive model and a novel architecture for managing relationships between the adaptive domains making up next-generation CSP IT architectures. In the past management of inter-domain relationships has only focused on centralised and static solutions. In practice this has produced brittle, limited, expensive and non-standard SDP integrations. Even when dynamic aspects were considered, e.g. with trader-style matchmaking services, the focus was on establishing the appropriate relationship rather than managing and maintaining a relationship through its lifecycle. As business models and service production becomes more flexible the importance of supporting this dynamic approach will increase.

References


