TrustCoM

Deliverable

16

The Trustcom Conceptual Models V1

AL1 Framework Definition

Lutz Schubert (HLRS), Michael Wilson (CCLRC), Jochen Haller (SAP), Alvaro Arenas (CCLRC), Adomas Svirskas (CCLRC), Pablo Giambiagi (SICS), Jürgen Doser (ETH), Emil Lupo (IC)

July 2005

Release V6

TrustCoM

A trust and Contract Management framework enabling secure collaborative business processing in on-demand created, self-managed, scalable, and highly dynamic Virtual Organisations

SIXTH FRAMEWORK PROGRAMME

PRIORITY IST-2002-2.3.1.9

Prepared for the European Commission under FP6 Contract No. 01945 as a deliverable form

Deliverable datasheet

Project acronym: TrustCoM

Project full title: *A trust and Contract Management framework enabling secure collaborative business processing in on-demand created, self-managed, scalable, and highly dynamic Virtual Organisations*

Action Line: AL1

Activity: Framework Definition

Work Package: all AL1 WP

Task:

Document title: The Trustcom Conceptual Models V1

Version: V6

Document reference:

Official delivery date: 31 July 2005

Actual publication date: 31 September 2005

File name:

Type of document: Report

Nature:

Authors: Lutz Schubert (HLRS), Michael Wilson (CCLRC), Jochen Haller (SAP), Alvaro Arenas (CCLRC), Adomas Svirskas (CCLRC), Pablo Giambiagi (SICS), Jürgen Doser (ETH), Emil Lupo (IC)

Reviewers: SAP, HLRS

Approved by: Michael Wilson

Version	Date	Sections Affected
V.1	05/07/04	Template for all contributions
V2	21/07/05	Awaiting CCLRC start & end and ETH section.
V3	25/07/05	Complete, for comments from all authors
V4	1/08/05	Comments incorporated, for internal project review
V5	26/08/05	updated after internal review

Executive Summary

The objectives of the Trustcom project are to produce a Framework, an architecture for implementing it, generic methods and tools as a toolkit to use the framework, and demonstrations of its applications. The Framework is built upon conceptual models which have been incorporated into the designs for the architecture, as well as upon the generic methods and tools identified. This document presents those underlying conceptual models for the following components:

- Virtual Organisation Management
- Business Process Modelling
- Contract and Service Level Agreement Management
- Trust and Security Management
- Policy Representation, Deployment and Monitoring
- Enterprise Network and Virtual Organisation Infrastructure

The conceptual models are generated through cycles of the following stages:

- Define initial conceptual models
- Conceptual models incorporated in UML domain models and activity diagrams
- Design modelling describes classes for each service
- Profiling applies to each WS* to state the service operations, the message exchange and protocols

This report presents the results of the first cycle through this process. The profiles resulting from the conceptual models in the final phase of each cycle are reported elsewhere as part of the Trustcom Framework for Contract and Trust Management.

Table of Content

Table of Figures and Tables

1 Introduction

The TrustCoM project [http://www.eu-trustcom.com/] is developing a framework for trust, security, and contract management, for secure, collaborative business processing and resource sharing in dynamically-evolving Virtual Organisations. The term "TrustCoM Framework" stands for the principles and paradigms, the processes and functions, and the architecture and the technology that underpin trustworthy, secure, and contract-driven operations of Virtual Organisations.

The Trustcom Framework includes the following components:

- a set of semantically well-founded **concepts** and relationships for describing and reasoning about trust and security in dynamic virtual organisations. This forms the meta-model of the TrustCoM framework;
- an abstract **architecture** reflecting these concepts, and providing a flexible structure and organising principles for systems based on the framework;
- **specifications** extending existing or defining new interoperability standards of services and protocols. (New interoperability standards will be defined when existing approaches can not provide an extensible basis to support the TrustCoM framework).

This document presents the first version of the first of these components of the Trustcom Framework – the conceptual models. This document will be updated at six monthly intervals for two more versions as experience increases of developing the architecture and specification components of the framework, and in implementing a generic toolkit to instantiate the framework, and then applying that generic toolkit to application scenarios to evaluate its effectiveness.

This introduction continues with a brief summary of the vision of Virtual Organisations captured by these conceptual models in order to introduce the core terminology, then describes the role of conceptual models in more detail.

Subsequent sections of this document then describe the models for different aspects of the scenarios:

- Virtual Organisation management
- Business Process Modelling
- Contract and Service Level Agreement Management
- Trust and Security Management
- Representation, Deployment and Monitoring of Policies
- Enterprise Network and Virtual Organisation Infrastructure

1.1 The Vision

The Trustcom project addresses the issue of trust within dynamic virtual organisations created, operated and existing within the Web or GRID. Virtual organisations (VOs) are created when consortia of legal entities wish to work together to produce a product, provide a service or tender for a contract, but do not wish to either have one contracted party to which the others are subcontracted, or to create a new legal entity which they jointly own. VOs can be created quickly, and undertake their role for a very brief period of time, or exist for the longer term. VOs can be established by a consortium or VO agreement which outlines the legal framework for the co-operation, within which specific Service Level Agreements can be produced to detail each service provided by a legal entity within the VO.

When a VO is formed its creator has a view of the business it is conducting, and the roles in the business process that need to be manned by the (potential) VO member. Therefore the VO must discover potential members who can meet these role definitions, negotiate their agreement to the VO agreement, and the individual role-reated SLAs. Within Trustcom, it is assumed that the descriptions of potential members, and the services to be provided are all available as Web or Grid services, open to automated resource discovery, negotiation and SLA agreement. One of the factors involved in the selection of VO members will be their previous reputations both to undertake roles defined in the VO business process model, but also to operate under VO agreements and SLAs, and even, their litigiousness.

Once the VO is operating, each participant must open up its internal ICT infrastructure to the other VO members in as much as they require it to undertake their roles in the VO. Consequently there are security issues for each VO member concerning the authenticated identity of employees of their own and other VO members, as well as authorisation issues of access to data and services throughout each organisation. While a VO is operating, the set of Web or GRID services brought together to achieve its business process model must be monitored and managed to ensure that the cost, time, security and quality measures stated in each SLA are being met, and when they are not, that appropriate actions are taken. The VO software embodying the resolution of these issues must also resolve the standard tradeoffs of distributed computing between orchestration and choreography as methods of service composition. The VO software must conform to open standards to permit the software interoperability required for disparate organisations themselves to interoperate as required to bring about dynamic virtual organisations.

When a VO has completed its activity, it must then be terminated to minimise future risks from liability and exposure to security breaches, and ensure the appropriate accounting of expenditure and income and distribution of profit or loss between the participating organisations.

This vision encompasses the business process model (BPM) of a VO, the roles defined in it, the discovery of organisations to fulfil those roles on the basis of published capability and reputation, the legal agreements between selected organisations to fulfil their roles, the establishment of secure and reliable composition of Web or GRID services to enact the business process models, the monitoring and management of the performance of members, and the decomposition of the VO resulting in minimal outstanding risks. The relation between the executable BPM, SLA management and security concerns all result in policies which must be deployed and monitored during performance to reduce risk within an international legal framework. Therefore each of these components BPM, discovery and negotiation, reputation management, SLA management, security policies, legal framework needs to be analysed, modelled and incorporated into a software system to host dynamic virtual organisations where risks and the trust required to offset them can be quantified and judged by business decision makers.

1.2 The Role of Conceptual Models

"A [framework's] conceptual model is the "mental map" that the designers build into it in order to present information logically. ..., the conceptual model represents how they expect and believe the information they are encountering should, and does, fit together. A hallmark of usable design is when both the designer's conceptual model and the users' conceptual models are in alignment."
[http://www.air.org/concord/wai/conceptual.html]

The conceptual models are generated through cycles of the following stages:

- Define initial conceptual models
- Conceptual models incorporated in UML domain models and activity diagrams
- Design modelling describes classes and interfaces for each service
- Profiling applies to each WS* to state the service operations, the message exchange and protocols

This report presents the results of the first cycle through this process. The profiles resulting from the conceptual models in the final phase of each cycle are reported elsewhere as part of the Trustcom Framework for Contract and Trust and Security Management.

The overall conceptual model for Trustcom following the vision is that:

1) a business decision maker evaluates the strategic and operational risks of a business vision, considers the resulting investment risks and the availability of the resources to enact the business idea and decides that a VO is the preferred form of entity to create. The business models relating to this decision and the enactment of the VO are described elsewhere in public deliverable DA on VO business models. The general concepts of a VO and its lifecycle are defined in section 2 of this deliverable.
2) To create a VO, various legal issues need to be considered before drafting a General VO Agreement (GVOA) to be signed by all partners. The legal background to this is described in deliverable D15 on legal issues.
3) The VO manager must now define the business process in a model (BPM). The conceptual model for this activity, and the later enactment of that VO BPM are described in section 3 of this deliverable.
4) The VO manager must now attract partners, negotiate the contract and associated Service Level Agreements (SLA) for each service. The conceptual model for these stages are defined in 4 of the current deliverable.
5) The GVOA provides the attachment of the non-functional requirements for the VO to the functional specification of the business process (BP), where these non-functional requirements are expressed as SLAs, trust and security policies. Other issues associated with Trust include reputation management which is used both when selecting VO members, and during the operation of the VO to monitor partner

behaviour. The conceptual models for these issues are described in section 5 of the current deliverable.

6) The derived policies for security, timeliness and quality must be enforced during the VO operation, and breaches in them reported for action. The conceptual models relating to policy definition and enforcement are defined in section 6 of the current document.

7) To support the implementation of these conceptual models is an infrastructure which has the concepts related to it described in section 7 of the current document.

Together these conceptual models provide a way of considering trust, security and contract management issues related to virtual organizations. Once created, the models are used to develop an architecture, and generic software specifications which are also reported elsewhere in the appropriate deliverables. The experience derived from developing the architecture and software then feeds back to improve the conceptual models.

2 Virtual Organisation Management (CCLRC)

Within Trustcom a Virtual Organisation is a union of organisations or individuals bound together by agreement to achieve an objective, or to exist for a set period of time. VO contrast with single companies, partnerships between companies, and companies contractually related by sub-contract relationships, in the following ways:

1) When somebody has an innovatory business idea, and plan to bring it to fruition, they could undertake the whole activity within a single organisation. However, this would require that organisation to both assume all the risks related to the enterprise, and to have available all the required resources.

2) If all the resources are not available, but the organisation wishes to assume most of the risk, then sub-contract relationships can be made to obtain the additional resources, or funds can be raised (e.g. equity, venture capital, loans etc) to provide financial resources. Some of the operational risks are usually passed to the subcontractor, but these are usually minor in respect of the whole enterprise.

3) If the organisation wishes to share not only operational risks, but also strategic risks then a longer term partnership relationship can be established through partial, or incomplete contracts which allow two organisations to work together. This relationship is becoming common between organisations and their IT providers, by acknowledging that the IT requirements to support changing organisational structures are rarely definable at the outset of an enterprise, so that risks associated with strategic direction can be shared as well as operational risks of completion of sub-contracts to time and budget.

4) The virtual organisation is a step beyond the partnership relationship in which two or more entities join together to achieve an enterprise. The relationship between them will be governed by a legal contract, or VO agreement. The options and details of what this agreement addresses will be described below, as will the life cycle of such VO.

The choice as to which basis to use to create and enterprise is complex, including identification and quantification of risks, the balancing of funding with those risks and trading off risks against each other. For example, a single organisation absorbs all the risk itself, but this entails maintaining both strategic and operational control over the enterprise. Therefore, when identified risks occur, the organisation is in control of all the prioritisation involved in recruiting resources to one activity at the cost of others. Alternatively, if an external organisation has responsibility for some functions (through sub-contracting, partnership or VO arrangements) then, not only is exposure to some of the risks passed to another organisation, but also the management of actions to overcome those risks when they occur. Consequently, when a sub-contractor is squeezed for resources, they may decide that failure to meet one contract with smaller penalty clauses, or where there is no long term business relationship to foster, is preferable to failing to meet another contract with more significant penalty clauses or business relationships. The move to outsource some functions involves a change from operational risk management to the management of risks associated with the outsourcing contract and its delivery.

Between the Trustcom partners, three scenarios of VO are illustrative of different business opportunities:

1) In the aerospace industry, engineering is undertaken between a large number of organisations, historically with a large prime contractor and a series of sub-contract relationships for specialist providers. The industry is moving towards both automating these existing relationships through IT, and in moving towards the VO as a contractual basis.

2) In the scientific community, large projects (e.g. the CERN LHC) are collaborations of 100's of organisations to design, build, and use experiments, as well as to analyse and publish the results. Currently, these collaborations are run as Grid mediated VOs (e.g. EGEE). However, there is little direct financial benefit from these VO, and consequently, very simple contracts and SLA, and very little security. The management of the VO consists of registering the certified individuals who are members and ensuring that there certificates have not been published on a black list of those who have broken the acceptable use policy for the Grid. The community intends to increase the financial, contract and security aspects of the existing infrastructure so that more commercial work can be undertaken within it.

3) The formation of new VOs as enterprises to provide innovative functions in a knowledge based economy will require an infrastructure that can itself be a service which can be commercially developed.

Each of these business opportunities require contractual arrangements to control the interoperation of resources in VO partner's own IT systems, and the security mechanisms to be enforced to ensure that they are conformed to. However, contracts, certificates and access controls are what social scientists term "trust substitutes". If the relationship between the VO members were sufficiently trusting, and the VO members were sufficiently trustworthy, then such measures would not be needed to substitute for that trust. Business experience does not suggest that trust should be relied upon over the long term between organisations. However, the more trust substitutes that are introduced and relied upon, the less opportunity there is for trust to develop between either organisations, or individuals. Since there will always be residual risk which has not been foreseen, there will always be a place for trust as the only mitigation. Consequently, the balance between managing risk through enforceable trust substitutes and encouraging trust, must be maintained within the Trustcom Framework.

The core contribution of the Trustcom framework is the ability to define an agreement/ contract at a business level and have it specified, monitored and updated at a technical, operational level. This refinement and implementation is done through the derivation of policies from the contact that can be enforced at decision points within the software architecture. The innovation in the framework is to take advantage of recent research into the representation, and enforcement of such policies to achieve this.

The next sections describe the conceptual models used in Trustcom for the VO itself and its evolution throughout its lifecycle.

2.1 General VO Model

This model assumes that the VO is created by instantiating a general VO agreement in the form of a common template specification (that E. Lupu et al. have called *the doctrine* in previous work done outside of the TrustCoM project). This template may be made available by one of the possible partners or by an external party who identifies the business opportunity but does not wish to exploit it herself. This approach simply suggests that a VO is not formed entirely through negotiation but some elements have to be specified in advance. The VO agreement addresses issues of the legal relationship and liabilities of the organisations in the VO, the starting and terminating conditions, the IPR relationship between VO members, and the objective of the VO. Addenda to the agreement will define the business process to be undertaken to achieve those objectives, the resources required to achieve those objectives, and the role each VO member will take within that business process. The template can be instantiated with specific details of the objectives, business process, liabilities etc. multiple times to create different VOs. The parameters that have to be provided upon instantiation are those elements that may be derived dynamically. The present design assumes that instances of services and actors are assigned to the roles dynamically and that QoS parameters for the SLAs may also be determined upon instantiation.

The information model[1] of the VO specification is shown in the figure below. It is divided into two levels of abstraction the "business" level and the "operational" level. Elements in the latter are meant to be directly enforced by an automated system whereas at business level they have a descriptive value. The advantages of this approach are twofold: i) they permit a purely business (contractual) view on the VO without any considerations of its enactment and ii) we preserve information about those elements that are not automatically enacted. When at a later stage we define and implement the means to enact them they can be refined at the operational level.

At the business level, a VO has an *objective* (for the moment specified as an attribute) and defines a *business process* and a *contract in the form of a General VO Agreement*. The *General VO Agreement* (GVOA) has a duration (as well as other attributes e.g., start date, end date, etc.) and specifies a number of procedures such as formation, dissolution of the VO as well as procedures for adding and removing partners. Each GVOA will be accompanied by a series of Service Level Agreements (SLA) that define parameters of each service, for the benefit of both the provider and the recipient. Later on we can refine these concepts to the operational view e.g., by defining these procedures as operational business processes that are enforced by the VO administrative roles. The business process has a number of *tasks* that must be fulfilled by a number of *roles – a role will provide a service*. In this view, the roles are considered to be filled by organisations. So an example of a business process at this level would be: construct airplane by manufacturing fuselage, and engine and then assembling them. The roles for this task would be respectively fulfilled by a fuselage manufacturer, a wing manufacturer and an assembly organisation. It does not detail which operational services and actor roles these tasks entail.

[1] Information used to perform tasks is organized or structured to allow disparate groups of people to use it. An information model is a model or representation of the details required by people working within a particular domain. An information model requires a set of legal statement types or syntax to capture the representation, and a collection of actual expressions necessary to manage common aspects of the domain.

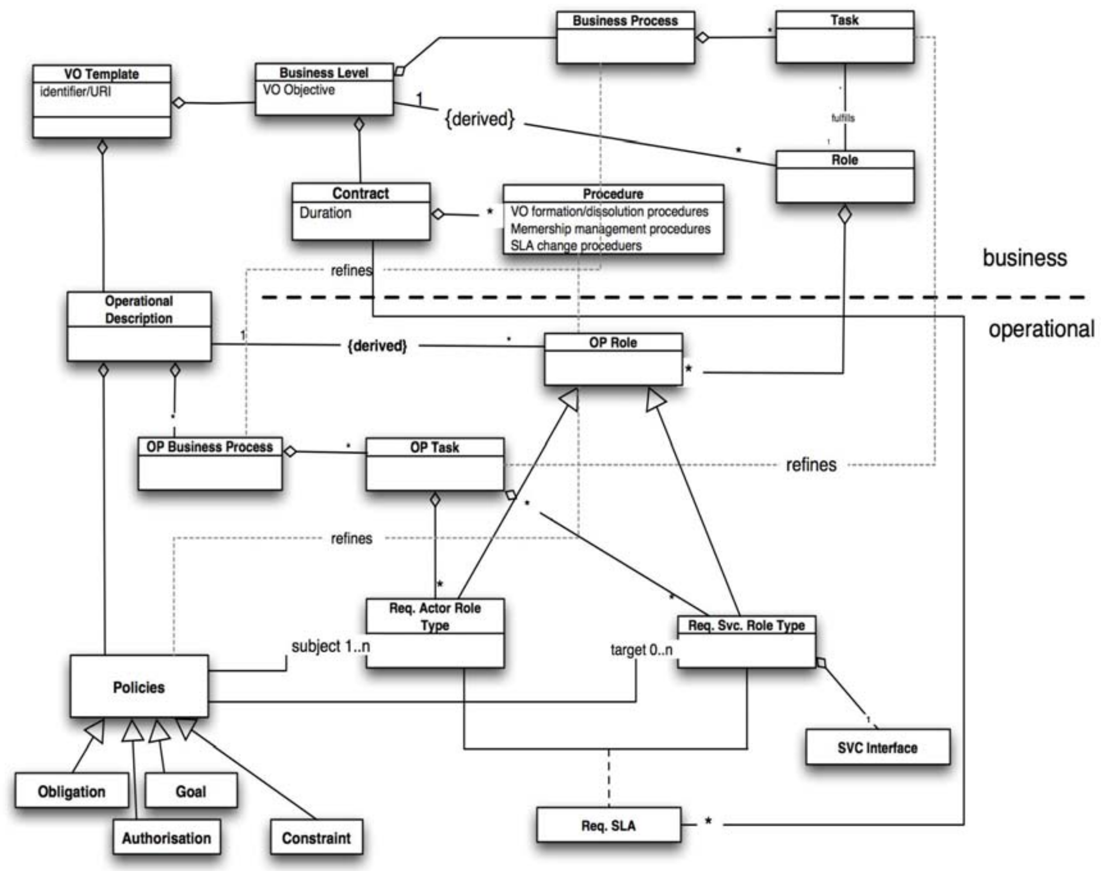

Figure 1 - VO Specification showing business and operational levels.

At the operational level a VO defines a number of *operational business processes*, which are a refinement of the business processes specified at the business level. For the moment we will assume that there is only one business process (arbitrarily complicated) which fulfils the objective. An *operational business process* comprises a set of *operational tasks* that may have **requirements** for either *actor roles* (i.e., objects initiating actions) or *service roles*, to which a particular web-service will be assigned, or both. Note that these role types can have multiple instances within the same VO. A service role comprises the service interface definition and the *SLA* template characterising the service. The latter contains the QoS parameters that are subject to negotiation upon creation of a new service role instance. Last but not least the operational description of the general VO agreement specification contains a set of policies that identify the *subject* and possibly the *target* to which they apply. The subject is mandatory as it is expected that all forms of policy must be implemented by some entity in the system. However, the target may be optional in that some types of policies may not require this in their specification. Four different types of policies can be distinguished: obligation policies that take the traditional event-condition-action rule form, authorisation policies that take the traditional XACML or Ponder form, goals that direct subjects to decide based on the strategy leading to fulfilment of the goal and constraints on behaviour.

Constraints may also include constraints on the instantiation of roles within the VO. There are risks associated with the reliance of a VO upon single points of failure – allowing a single organisation to undertake all the instances of one role within the VO, without any backup; or allowing a single organisation to undertake multiple roles spread randomly across the VO, so that if it fails the whole VO will fail. In this case a constraint may be introduced that no more than X roles of a given type may be instantiated by a single entity, or that the same entity cannot be assigned to more than X different roles.. Such constraints could apply to the VO membership manager at the time that members of the VO are being selected. These constraints would reduce a class of risk in VO operation due to organisational failure. However, again there is a trade off between risk management in the VO, and the risk of the technology failing. If the VO management system includes such constraints, is its risk of failure increased more or less than the reduction in risk to the VO of organisational failure ? At present Trustcom has chosen not to introduce such complex constraints favouring the mitigation of technical risks, but these may be introduced at a later stage when those risks are reduced.

Dynamic aspects. When new entities join the VO, new instances of the role types defined in the specification are created. Note that the SLA is associated with the Service Role Type and therefore each service role instance may have different SLAs with different QoS parameters. Partners can leave the VO at run-time. This is achieved by deleting the role instances corresponding to the actors and services provided by that partner. If an instance of an OP task ends up with no associated instance roles this implies that the task cannot be achieved. Depending on the business process and the GVOA this may or may not compromise the existence of the VO.

When partners are to be identified to fulfil a role in the VO, either a closed list of potential partners must be searched, or an advert published to an open group of potential partners – or a combination of the two. An optimal creation process would include all possible potential partners, and therefore would not rely on closed lists. However, completing replies to every advert in a different format is time and resource consuming for potentially little return. On the other hand, when potential partners register on a closed list they can register the services that they offer to a wide range of potential VO, consuming fewer resources and causing less delay in the formation process. There is a trade-off here between a slow process that is resource consuming for the potential applicants, but open to a wide set of applications, against a faster process that is less resource consuming, but closed to a limited set of applicants.

There is no perfect resolution to this trade-off. The introduction of a standard advert reply format that could be written once, thereby saving resources, may initially seem perfect. However, that has the same form as a registry where organisations are registered. Consequently, Trustcom has chosen to use the registry of potential VO members as a solution. The concept of the Enterprise Network is used in Trustcom to describe this list of potential partners. An Enterprise Network is a list of those entities that have registered as potential members of VOs.

The long term problem of how to attract potential VO members to the Trustcom registry as opposed to any other is only addressed by making minimum requirements on what needs to be placed in the registry. No specification is made of an advertising process. This would appear to be a low cost activity, and will be easier to make consistent with competitors as the set of VO formation mechanisms becomes more mature.

Once potential VO members have been discovered, some form of negotiation is required of the contract and SLA before they can join a VO. It is possible to perform the negotiation in two stages, firstly a business negotiation to ensure that the required service can be provided at a price, and then a technical negotiation to define the details of the service provision in the SLA. However, this sequence would result in a technical negotiation with a party who already knows that it is the only bidder. Unless there is considerable trust between the parties, this negotiation will be very one sided in favour of the provider. Alternatively, the negotiation could take place in parallel for the two stages (business and technical) which would improve the bargain made for the VO initiator, but would vastly increase the complexity of the negotiation process. The details of the negotiation process itself are described in a later section, but at this stage it is necessary to understand the trade-off from the perspective of managing the formation of the VO.

The following diagram contributes a refinement of the VO Specification at the operational level.

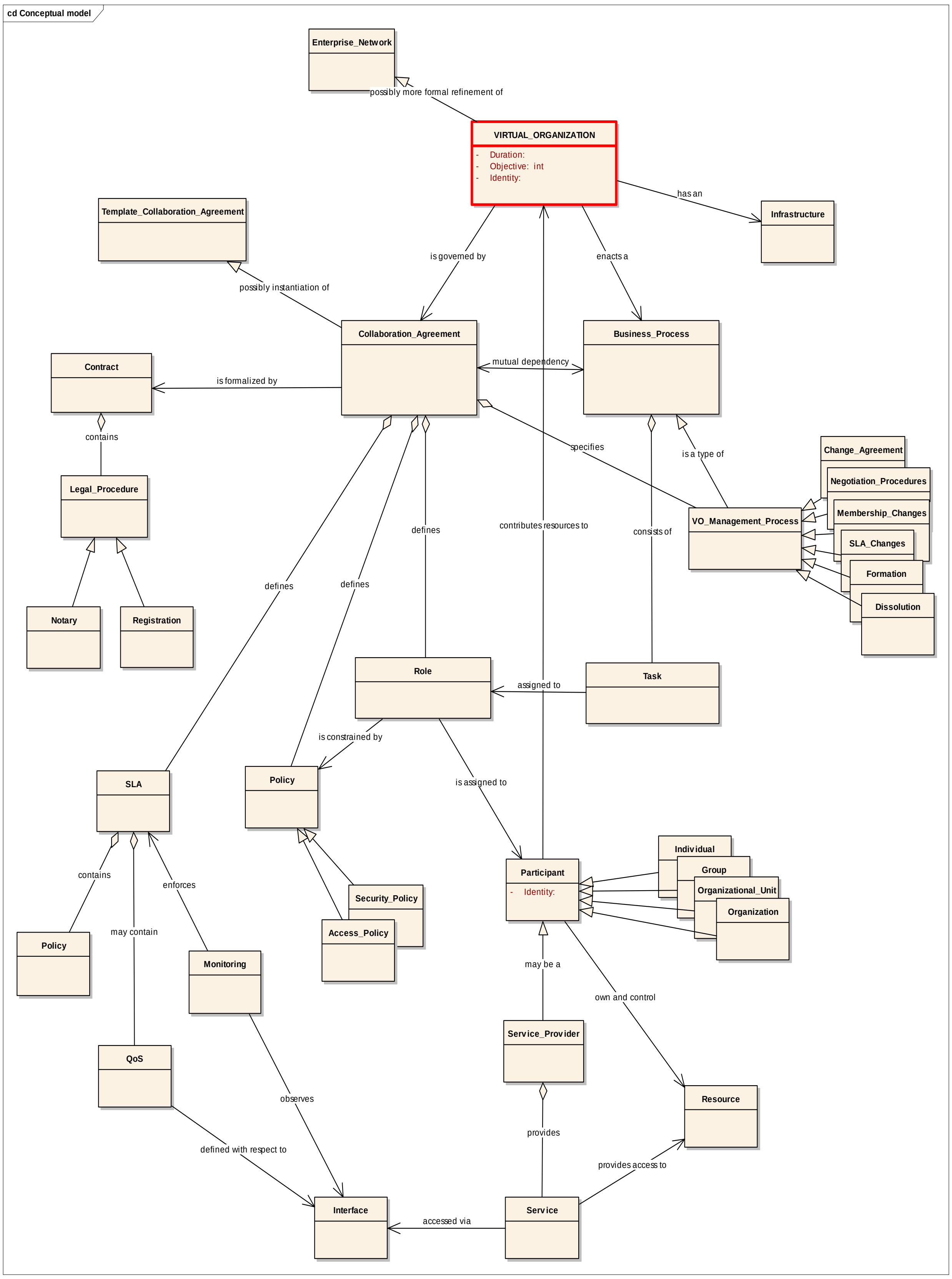

Figure 2 - VO Specification (Operational Level)

2.2 VO Lifecycle

TrustCoM is following the life-cycle model developed in the VO roadmap project (Camarinha-Matos and Afsarnabesh, 2003), including the phases of identification, formation, operation/evolution and dissolution. The identification phase covers setting up the VO; this includes selection of potential business partners by using search engines or looking up registries. VO formation deals with partnership formation, including the VO configuration by a VO Manager, who distributes information such as policies, Service Level Agreements (SLAs), etc, and the binding of the selected candidate partners into the actual VO. After the formation phase, the VO can be considered to be ready to enter the operation phase where the identified and properly configured VO members perform accordingly to their role. Membership and structure of VOs may evolve over time in response to changes of objectives or to adapt to new opportunities in the business environment. Finally, the dissolution phase is initiated when the objectives of the VO has been fulfilled. Trustcom has added a sub-phase to this last to account for final Termination of the VO in which final liabilities are terminated.

There is also another stage added prior to the VO identification, in which the Enterprise network is established. This provides the set of candidate members for any VO.

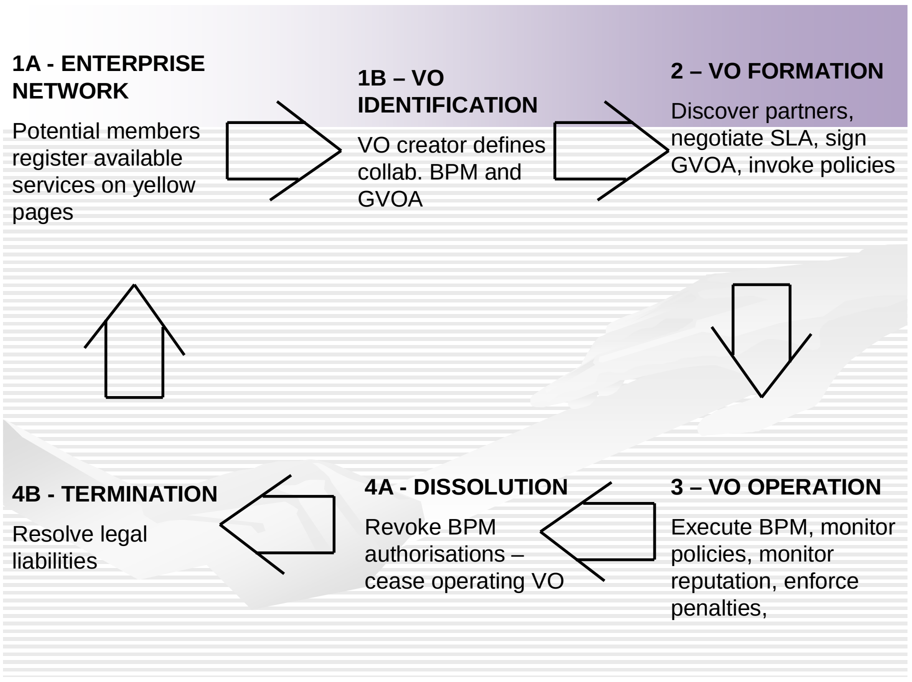
1A - ENTERPRISE NETWORK
Potential members register available services on yellow pages
1B – VO IDENTIFICATION
VO creator defines collab. BPM and GVOA
2 – VO FORMATION
Discover partners, negotiate SLA, sign GVOA, invoke policies
3 – VO OPERATION
Execute BPM, monitor policies, monitor reputation, enforce penalties,
4A - DISSOLUTION
Revoke BPM authorisations – cease operating VO
4B - TERMINATION
Resolve legal liabilities

Figure 3: The VO Lifecycle

In the rest of this section we analyse these phases, identifying the main challenges from the trust and security perspective and explaining the main concepts..

2.2.1 EN Creation and VO Identification

The enterprise network creation involves establishing the Trustcom EN Infrstructure and allowing organisation to register their interest in potential VO.

The identification phase includes defining a Collaboration Definition, where the VO business objective and its corresponding roles are defined, as well as trust, security and contract management (TSC) properties associated to the roles and their interaction. The roles and their TSC properties are used as the base for discovering potential business partners from the EN members who are both capable of fulfilling the required roles and of fulfilling the TSC requirements of the VO by using search engines and/or looking up registries.

2.2.2 VO Formation

During the formation phase the selected set of Members needs to be limited to those who will actually fulfil the roles in the VO, and configured so that they can perform according to their anticipated role in the VO.TSC properties are refined into policies. An important document generated in this phase is the General VO Agreement (GVOA) which record the VO policies as well as the Service Level Agreement (SLA) associated to the services provided by a partners. SLA will be negotiated with each VO member for each service provided.

2.2.3 VO Operation

This phase can be considered as the main life-cycle phase of a VO. During this phase the identified partners contribute to the actual execution of the VO tasks by executing pre-defined business processes. Important features in this phase are the monitoring of the performance of the VO as well as the enforcement of policies.

2.2.4 VO Evolution

VO Evolution is part of the VO Operation phase. When a VO member fails completely or behaves inappropriately, the VO manager may need to dynamically replace such partner. This evolution may involve discovering new business partners, re-negotiating terms and providing configuration information, as done in the identification and formation phases. One of the main problems involved with evolution consists in re-configuring the existing VO structure so as to seamlessly integrate a new partner, possibly even unnoticed by other participants.

2.2.5 VO Dissolution and Termination

The dissolution phase is carried when the objectives of the VO has been fulfilled. During dissolution, the VO structure is dissolved and final operations are performed to annul all contractual binding of the partners. From a trust and security perspective, this involves resolving federations, revoking security credentials, invalidating VO context information, and updating reputation of all participants.

The final termination of the VO may take place many years after the dissolution since some liabilities may persist after the VO has dissolved. Therefore records must be maintained of the VO membership, in case such liabilities need to be resolved.

2.3 VO Management

Current Web Services and Grid based VO management systems support a VO membership function alone – listing VO members who are entitled to use VO resources. They do not support the creation of a contractual basis for VO membership, monitor the performance of services by VO members to performance indicators established in those contracts and SLA's nor take action when the indicator levels contracted are breached.

Trustcom will provide a VO management system that supports membership, but also supports the high level contractual relationship, and its refinement down to monitorable policies, the monitoring of those policies, and notification of policy breaches. The concepts relating to these issues are mostly described in more detail in later sections.

One to consider here is that of the action to be taken by the VO management system when policies are broken. If a policy defined in a contract or SLA has been broken what action should the VO manager take ? Adaptation policies - also sometimes referred to as Obligation policies are in essence event-condition-action rules that determine how components should adapt in response to events arising as part of a VO. The SLA or contract may state penalty clauses that apply under such circumstances, if so, should such clauses be automatically enacted – should VO members be fined, or removed from the VO automatically ? Although it is technically feasible to do this, it is not believed that the market is yet ready for such dramatic automation. Rather it is assumed that the human who manages the VO should be notified and human action may be required. This is an assumption based on discussions with those involved in current VO, and other outsource relationships. As the technology matures and the market becomes more accustomed to it, the opportunity for increased automation in actions is expected to increase, but at present the more conservative option is judged to be appropriate for the market. However, the automatic action as a result of policy breach is still a research topic that will be investigated within the Trustcom Framework for events including the arrival of a new participant or departure of an existing one, failures, access control violations, changes of trust or reputation that exceed given thresholds.

3 Business Process Modelling (SAP)

While BPs in the architecture (WP27-D9) deals more with the BP subsystems operational baseline, component lifecycles and their interactions, e.g. how public and private BPs are derived from a collaboration definition (CD), the conceptual model deals with their corresponding classes and the trust security and contract management related BP control concept (or TSC concept for short) is introduced. The TSC concept fits into the baseline's class model and provides controls for the BP control flow based on decisions of the different TSC subsystems. The goal is to model the TSC concept on a level that it may be applied to different underlying BP models and methodologies and that it may be extended to work with other than only TrustCoM specific security related subsystems.

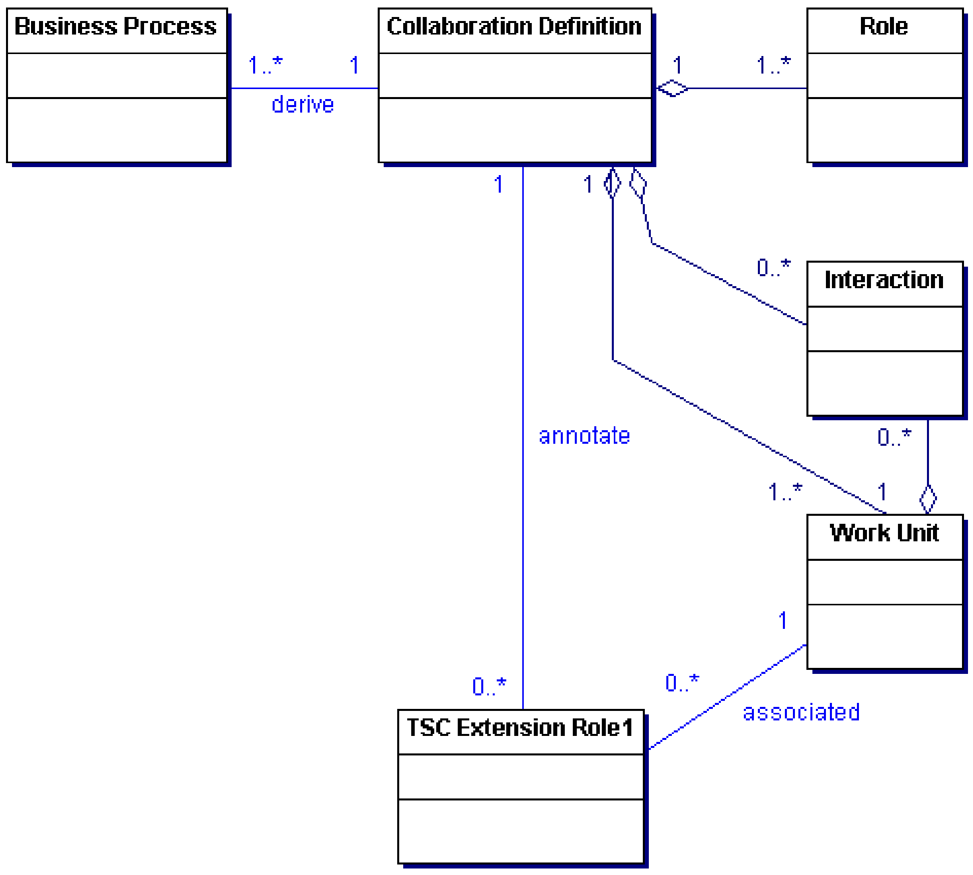

Figure 4: Design time BP classes

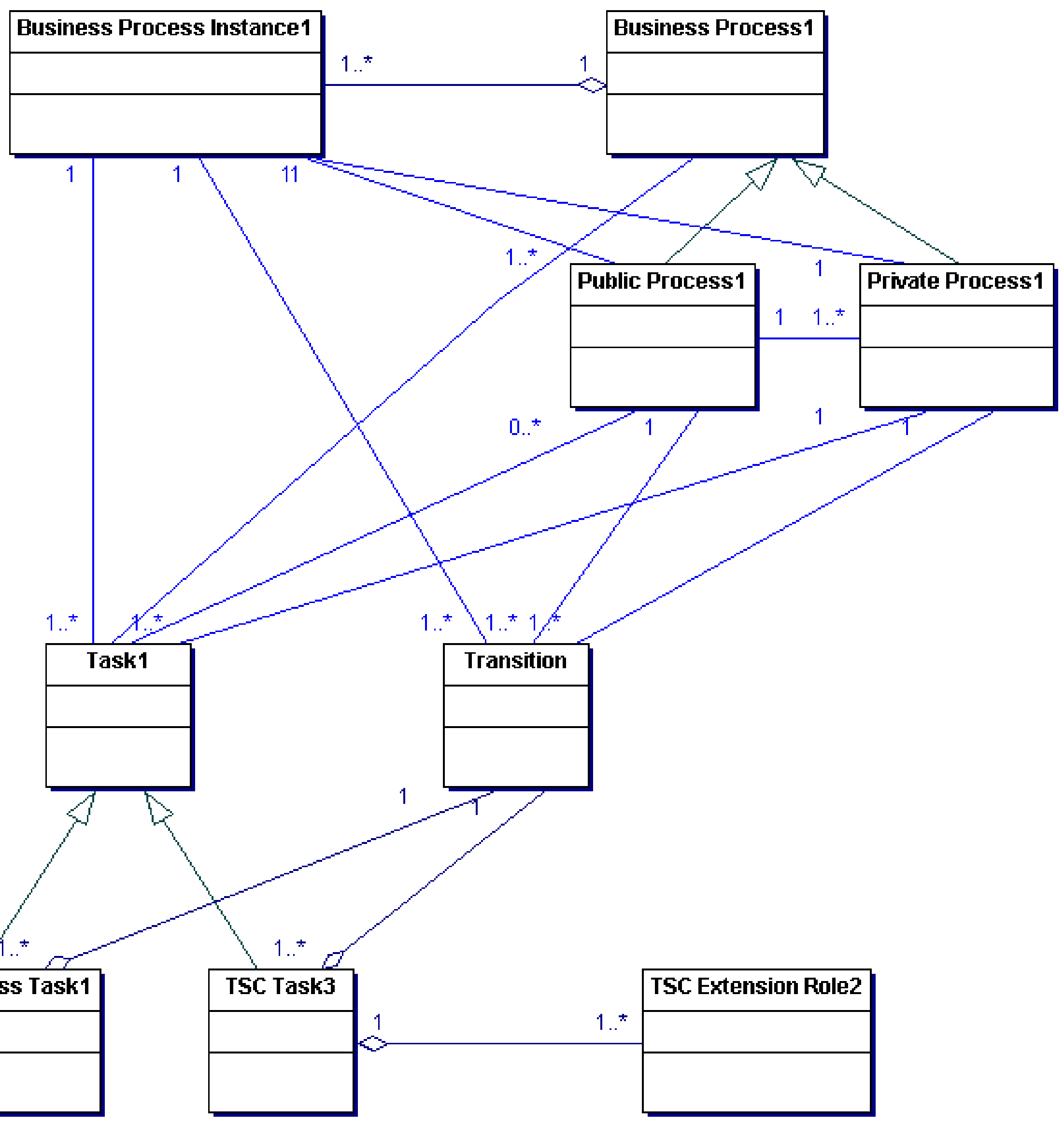

Figure 5: Runtime BP classes

Figure 4: Design time BP classes and **Error! Reference source not found.** illustrate the secure collaborative BP (CBP) classes, their relations and associations which deliver the baseline for the improved and refined secure collaborative business process models of the second phase in TrustCoM. The diagram is split into two parts to highlight the design and runtime aspects. Design time classes are relevant, as described later, for early VO phases (identification and formation), basically to derive instances of runtime classes for the operation phase at runtime.

The deployment model introduced in WP2 starts at BP design time with a collaboration definition (CD) which consists of the following parts:

- CD (or Operational) roles, which have to be filled with VO members.

- Work units, which describe business activities on a higher level. The description is not meant to directly map to service invocations yet, although a service EPR may be inserted if already possible. Rather, a concatenation of business keywords is given which determine required services for later business process derivation.
- Interactions among roles, which map to exchanges of business messages transmitting work unit relevant information and data. Business messages model the later expected sequence of message exchanges between processes and services.

TSC controls that are already determined at this point in time may be added to the collaboration definition as well. Such controls are specified in TSC extension roles, which are associated with a specific work unit and annotate the CD. This concept is described in 3.3 to the level of detail as currently possible. WP21's work in the next modelling cycle will focus on the TSC concept and in the end deliver a mature TSC model for BPs.

Still in design time, the CD is used as a basis to derive CBP's for each role. The TrustCoM Architecture (WP27, D9) describes the components and their interactions performing this derivation in detail. A CBP, as already introduced in WP2, ID1.1.5, consists of two parts:

- Private process: the executable process doing the actual work to meet a business objective
- Public process: the process exposed to the outside facilitating CD interactions with other roles in detailed message exchanges (can be perceived as a private process's interface to the outside of the own administrative domain)

A public process is supposed to hide sensitive, highly optimised private VO Member processes which should not be disclosed even to other VO Members.

All types of BPs have one class in common, the task. A task is the atomic business component encapsulating a piece of work or activity which can be performed internally by one VO Member, playing one CD role without interacting with another partner or role.

Two types of tasks are possible, business tasks and TSC tasks. Tasks may exclusively contain two types of activities:

- Business activity, an operational part of the work which contributes to the overall achievement of the business objective
- TSC task, further described in the following subsection, an activity dealing specifically with dynamic trust, security and contract management requirements based on the TrustCoM subsystems

Dynamic in this context means that TSC controls are introduced during CBP runtime. At this point in time, when the BPs are deployed design time turns to runtime upon start of BP execution. At least one private and public process instance is created for each BP. The general task and transition structure of each BP instance is identical with the design time BP model structure.

Task attributes are defined to contain at runtime workflow relevant data, which is processed by executing the BP instance. For a business activity, the attributes contain operational data, e.g. references to aircraft design data. TSC task attributes are configured and filled in by TSC extension roles. For TSC extension roles that are already contained in the CD, this leads to fully prepared and configured TSC tasks upon BP instance creation. To cater for dynamic CBP TSC controls, TSC extension roles may also be deployed to a VO Members BP execution environment during runtime just before the TSC Task which should be configured is executed. As a prerequisite, it has to be noted that TSC Tasks need to be present (without associated TSC Extension Roles) in the design time CBP model. The TSC Task is described in more detail in 3.5. A more detailed deployment model is presented in WP27, D9.

3.1 VO lifecycle

The BP subsystem intends to provide well specified services for other subsystems, catering for secure CBPs (cf. WP2, ID1.1.5) which are essential for the VO itself as well as the ones contributing to the operational, business goal of the VO. At least in one case, when the initial, overall CD of the VO is retrieved by VO Management and BP are derived from it for all required roles, the VO phases have to be considered.

3.1.1 Identification Phase

VO Management retrieves the initial CD from the BP repository (cf. WP 27 for the concrete BP component model).

Also in this phase, additional TSC Extension Roles may already be defined and added to the CD. If it is not yet possible to specify all TSC Extension Roles at design time, the TSC Extension role, this can be catched up on at runtime, before the TSC task is executed. It is still mandatory to denote the requirement of a TSC task in the CD at this point, by annotating with an empty TSC extension role. The CD is then passed for public process derivation to the BP subsystem.

3.1.2 Formation Phase

BP subsystem queries VO Membership Management for VO members meeting the specified roles from the CD. Upon received VO member list, the public and optionally also private processes can be derived for each role. The former are sent to the VO Members playing a certain role who deploy public (and private processes) in their BP execution system.

3.1.3 Operation Phase

The VO Initiator, the role which defined the overall VO Objective starts the enactment of the overall CBP to meet the objective. Either the initiator explicitly changes the state of the deployed process from inactive to running by using a a management service (the BPM service, cf. WP27, D9) or by sending the initial data in a message to the public process. Both possibilities depend on the process model, how the arrangement of tasks and transitions expect to begin the process execution. VO Members have already deployed public and private processes which are instantiated upon receipt of the first message in their execution system.

3.1.4 Dissolution Phase

Processes instances are destroyed in this phase, at least public processes, but also private ones of prescribed by VO agreement. With the end of the private and public process execution, the CD ends as well.

The concept of design and runtime for this overall CD can be mapped to the VP phases as well. From a BP perspective, identification and formation phase belong to the design time of the CBP, the operation phase marks the runtime of the CBP.

3.2 Secure CBP TSC Concept

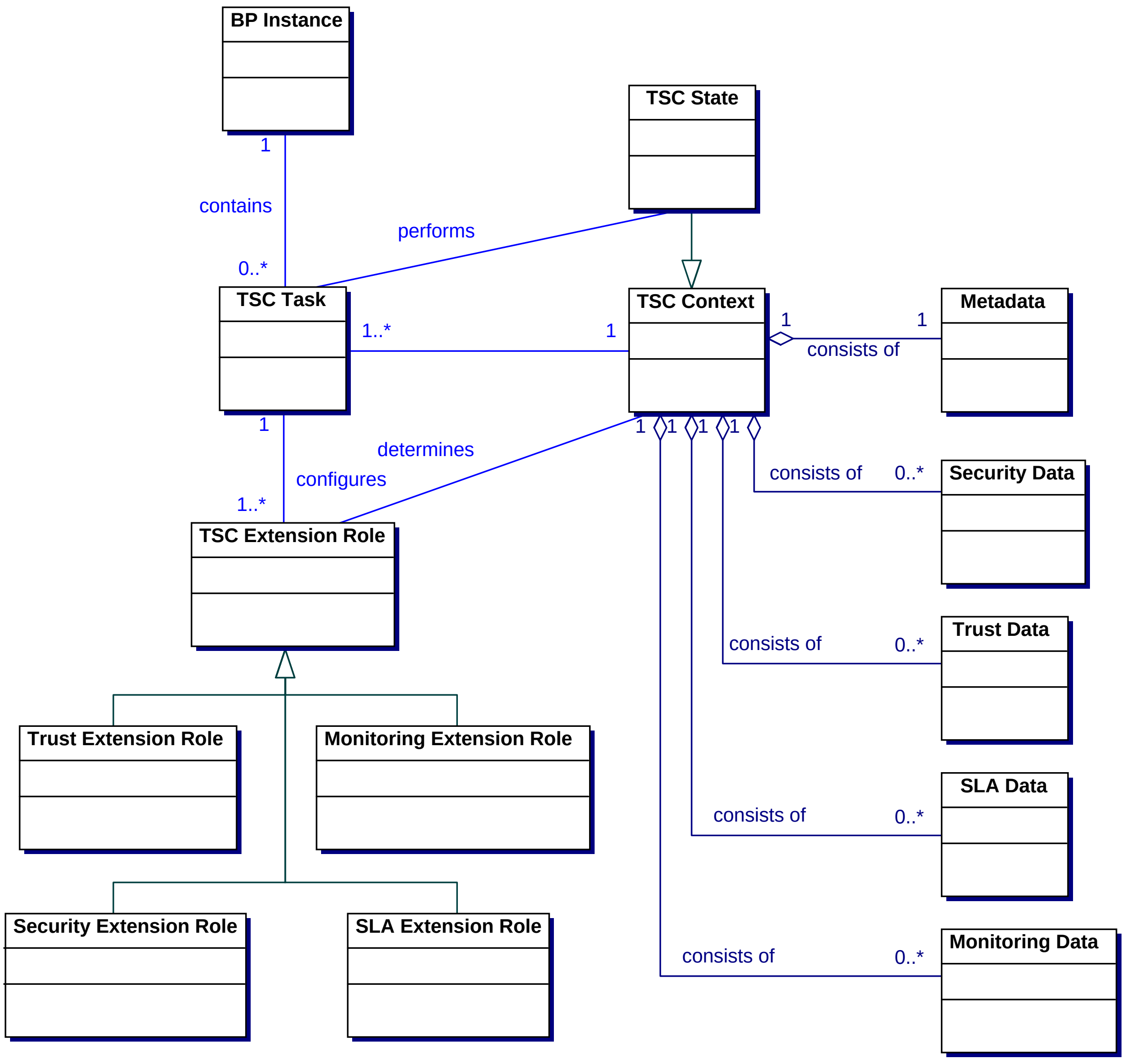

Figure 6: TSC Component Classes

Figure 6: TSC Component Classes jumps into the details of the TSC concept by extending Figure 4: Design time BP classes and Figure 5: Runtime BP classes, which provided the big picture, a wider overview of BP classes.

The TSC concept can be subdivided into three major components, which will be described in the following sections in more detail:

- TSC Extension Roles
- TSC Task
- TSC Context

The main contribution of the TSC concept can be summarized as bringing TSC controls from different other TrustCoM subsystems, e.g. SLA, Security and Policy, into design and runtime of secure CBPs.

3.3 TSC Extension Roles

TSC Extension roles capture all information which are necessary for TSC tasks to perform their duty. A TSC Extension Role is modelled as a data set containing all the required information to configure a TSC task. So far, four types of TSC Extension Roles are modelled, each realizing a CBP control for exactly one TSC subsystem:

- Trust Extension Role – Trust and Reputation subsystem
- Security Extension Role – security subsystem
- SLA Extension Role – SLA subsystem
- Monitoring Extension Role – Messaging Subsystem

The TSC role specification consists of:

- Metadata
    1. Name of the role
    2. Process ID for deployment
    3. TSC Task ID for runtime deployment
    4. other Metadata (author, timestamp etc.)
- Subsystem section (one for each extension role)
    1. References, EPRs to be accessed at execution time of the TSC task (e.g. the URI of the SLA evaluator service)
    2. Parameters for the service call (e.g. an SLA ID)
- Signature or similar integrity/non repudiation mechanism

Table 1 - TSC Extension Role contains the data model of a TSC extension role as available after TrustCoM phase 1. Especially the runtime data sets are gathered from AL2 designs and AL1 – WP27, D9 inputs.

The table first states the logical type of extension role entry, followed by the data type, the runtime configuration if applicable (e.g. the endpoint reference (EPR) of a service invocation), expected return values and a comprehensive description

	Type	Data (used as parameter during runtime)	Runtime configuration (URI, method)	Expected return	Description
Matadata	TSCRoleName	tscRole			Name of the TSC Extension Role
	ProcessID	Process_ID, [CD_ID, view_ID]			Filled in upon deployment, ID of the process instance the role belongs to
	TSCTaskID	tscTask_ID			Filled in upon deployment, ID of the TSC task the role configures
	Author	String			Author of the role
	CreationDate	date, time			A timestamp
Trust	ReputationService	Member_ID, reqReputation	EPR,getReputation	Reputation	Reputation of a member is checked, in case of an unmet threshold,a BP exception handler has to be invoked.
Security	Policy_Status	Policy_ID	Unclear at that point in time, probably the policy only has to be referenced in the GVOA	boolean	Check of a policy's enforcement state
	SecurityTokenService_TokenValidity	Token	EPR, validate<type>Token	{OK,REVOKED,INVALID}	Validation of BP specific security tokens (not regular EN/VO tokens)
C	Contract_Status	SLA_ID	EPR, getWholeStatus	boolean	Check of a SLA violation
Monitoring	Monitoring_ReadEntry	Log_ID, EntryNr/TimeDate	EPR, readEntry	EntryNr, TimeDate, Message	Check of a particular log entry
	Monitoring_SendNotification	BP_TSC_Notification	EPR, sendNotification		Emitting a notification, the EPR references the local notification proxy

Table 1 - TSC Extension Role

An Extension Role is not necessarily required to be specified manually by a user, extension role templates may be used which are complemented by service EPRs and invocation data (e.g. parameters needed to invoke an EPR method) prior deployment. In the early VO phases, the VO Management subsystem (VO Lifecycle Management maintaining and initialising the General VO Agreement or GVOA for short) is the most likely source of initial TSC Extension roles which will be already annotated at design time in the CD. On the one hand, BP classes which are instantiated in the process of setting up a CD and of interest for the VO in general are passively tracked by the GVOA. Those are basically the CD itself and derived public processes (see D9 for details), where the BP subsystem generates notification messages containing references to those instances which are received by the subscribed VO lifecycle management and consumed in order to maintain a VO state captured in the GVOA.

On the other hand, the BP subsystem will require instance references of other subsystems to perform BP controls defined by the TSC extension roles. An example would be to query for a certain policy reference denoting the policy instance behind the reference is deployed in a PDP and active.

Security, Trust and SLA TSC Extension roles executed at runtime by a TSC task require a TSC subsystem decision at runtime. The required subsystem specific object on which the decision is based is provided in the data field, the invoked endpoint and method is part of the runtime configuration field.

The following subsection runs through a small example which will be later extended when the TSC Task is described in more detail. The presented TSC control is specified by a SLA Extension Role:

As an example, in the process snapshot depicted in Figure 7 - TSC Task*, the operational task n is executed prior to task n+1 requiring additional control decisions in between provided by TSC subsystems. For instance, both task n+1 requires a particular SLA to be in place which has to be verified. With verification in terms of a SLA, it has to be verified if the SLA is violated or not. SLAs are not verified on the enterprise level, but on the service level by an SLA subsystem. The SLA evaluator service acts as access channel for TSC tasks on the process layer. The service accepts an SLA identifier, the ID, as input and returns the violation status of this SLA as a Boolean value. Depending on the return value, the process flow will continue in one of two possible directions, either task n+1 or task (n+1)' will be executed.*

If it is assumed that only the requirement for a SLA check during process execution is known at design time, but not the specific details, e.g. the ID of the SLA, the TSC task has to be configured accordingly during runtime. This is done by assigning a TSC extension role, in the example the role "SLA control" during process execution, but before the TSC Task which needs to be configured has been executed.

Following up on the deployment of a TSC Extension Role, the role document may be deployed into the business process management system by invoking a service interface. The TSC task and process IDs map it to a unique task in a specific process instance. Of course, assignment of a TSC extension role has to occur prior TSC task execution or else the executing engine raises an exception.

The following runtime steps mark the beginning of a more detailed TSC extension role deployment model:

1) TSC Extension Role has to be specified, technically a document will result
2) Assisted by the BPM service, assign a TSC extension role to a particular TSC task
3) the BPM service fills the metadata section, role name etc. in the TSC task's attributes (an internal data object), the actual data needed for later TSC task execution (e.g. the EPR and parameters) in the TSC context and puts references to those entries in TSC task attributes as well
4) At TSC task execution time, the BPM fetches the relevant data for the currently executed task ID from the TSC. The data is identified according the references from the previous step captured in the task's attributes; the data is stored temporarily in a TSC state in the execution space of the process instance.
5) The TSC task's duty according to role assignment is now performed, for instance to check for SLA violations which will influence the following tasks; the EPR and the parameters (SLA ID) were fetched in the previous step from the TSC context and are stored in the TSC state; the TSC task invokes the service at EPR with a given method and parameter <SLA_ID>.
6) The subsystem's service implementation or instance itself, e.g. the SLA subsystem, takes care of the decision; the service implementation knows the detailed information, the TSC task is not required to have that kind of knowledge, it only needs access to the service interface.
7) The decision, here SLA violated or not, is returned to the TSC task by the service implementation.
8) The TSC task is now able to control the further process flow, e.g. in case of a positive answer from the authorisation subsystem, a task n+1 is allowed to be executed, otherwise a task (n+1)' continues.

The TSC context introduced above will be specified in more detail in 3.4.

3.4 TSC Context

The TSC context is created together with a BP instance and logically linked to it during runtime. The entire TSC concept's main contribution is to provide secure storage external to the BP execution environment for all TSC task configurations, stemming from deployed TSC Extension Roles, of a process instance.

The context, in general empty in the beginning except for metadata (see table below), is partitioned according to subsystems, e.g. one section refers to security, another to SLA etc. The partitioning canonically reflects the TSC Extension Role model since those are modelled following the same idea. Actual data is filled in upon TSC extension role deployment which may occur at design time already, as annotated artefacts in a CD, but may also occur at runtime to take dynamic TSC requirements for CBP into account. The TSC context itself contains no confidential data, but references to subsystems which in turn handle confidential data.

Nevertheless, the TSC context needs to protect the integrity of such data, since modifications of TSC Task configurations would undermine the security concept of BP controls. For instance changing a reference of a SLA to another one which will not be violated by a misperforming VO Member would void the SLA BP control.

Following this line of thought, if an SLA control is modelled by inserting a TSC Task in the BP model, the associated TSC Extension Role must not contain the SLA itself (or a copy), but merely references the SLA. Since tampering of such references is considered critical as well, the TSC context is stored separate from the process execution space it refers to, e.g. in a database or similarly trusted, reliable storage subsystem, but accessible by the BPM system.

The TSC context is associated with exactly one process instance. Table 2: TSC context provides an initial specification. The TSC context resembles in many fields a TSC extension role since the TSC context values are filled by the set of all deployed TSC Extension role contents.

The table is structured as follows:

- A high-level contents classification in form of metadata
- A proposed API method name to access certain fields (will be offered as a service to the BP engine and BPM service)
- The TSC context field itself as deployed by a TSC Extension role; since such a TSC Extension role targets exactly one TSC Task, the relation is assured by inclusion of the correct TSCTask_ID
- The results of the API call
- A comprehensive description field

Category	API – Method	Field	Type	Description
	getProcessInstanceRef()	ProcessInstanceRef	Process_ID	Reference to current proces instance
	getCDRef()	CDRef	CD_ID	Reference vector to relate collaboration definition, view and private processes in th same domain; also deliver therefore own process type
	getTSCRef()	TSCRef[]	TSCTask_ID[]	Reference vector of all TS tasks in the process instanc
	getTSCExtensionRoles()	TSCExtensionRole_ID	TSCExtensionRole	Reference vector of alread deployed TSC Extensio Roles
	getMissingTSCExtensionRoles()	TSC_ID[]	TSC_ID[]	Vector containing Ids of TS Tasks lacking a TS Extension Role
Metadata	getCurrentTaskRef()	taskID	taskID	Reference to current executed task
Trust	ReputationServicegetPolState()	TSCTask_ID, Member_ID, reqReputationPolState	EPR,getReputationVector	ReputationVO Manageme Alert Vector for Polic Violations
	AuthorisationMode()	TSCTask_ID, processID	{TBAC, CAP}	Authorisation Model
	PartnerRoleProcess()	TSCTask_ID, processID	{Role}	Role under which th process instance is execute
	PartnerRoleTask()	TSCTask_ID, taskID[, processID]	{Role}	Role assigned to a particul task under which it executed
	PermCheck()	TSCTask_ID, taskID[, processID]	{permResult}	Check task permissions if th current task
	getPolicy_Status()	TSCTask_ID, Policy_ID	boolean	Verify the enforcement stat of a particular policy via tbd
Security	getTokenValidity ()	TSCTask_ID, Token_ID	boolean	Check validity of a securi token
Contract	getContract_Status ()	TSCTask_ID, SLA_ID	boolean	Check status/violation of SLA
	readEntry()	TSCTask_ID, Log_ID, EntryNr/TimeDate	Notification	Read a particular log entr based on ID and/or TimeDat
Monitoring	sendNotification()	TSCTask_ID, Notification, Topic	errno	Send a notification for particular Topic

Table 2: TSC context

The TSC context is comprised of exactly one metadata field section. The fields for the different TSC subsystem may occur multiple times, ranging from zero to the number of TSC Tasks modelled in the process instance, depending on the TSC Extension role configuration.

3.5 TSC Task

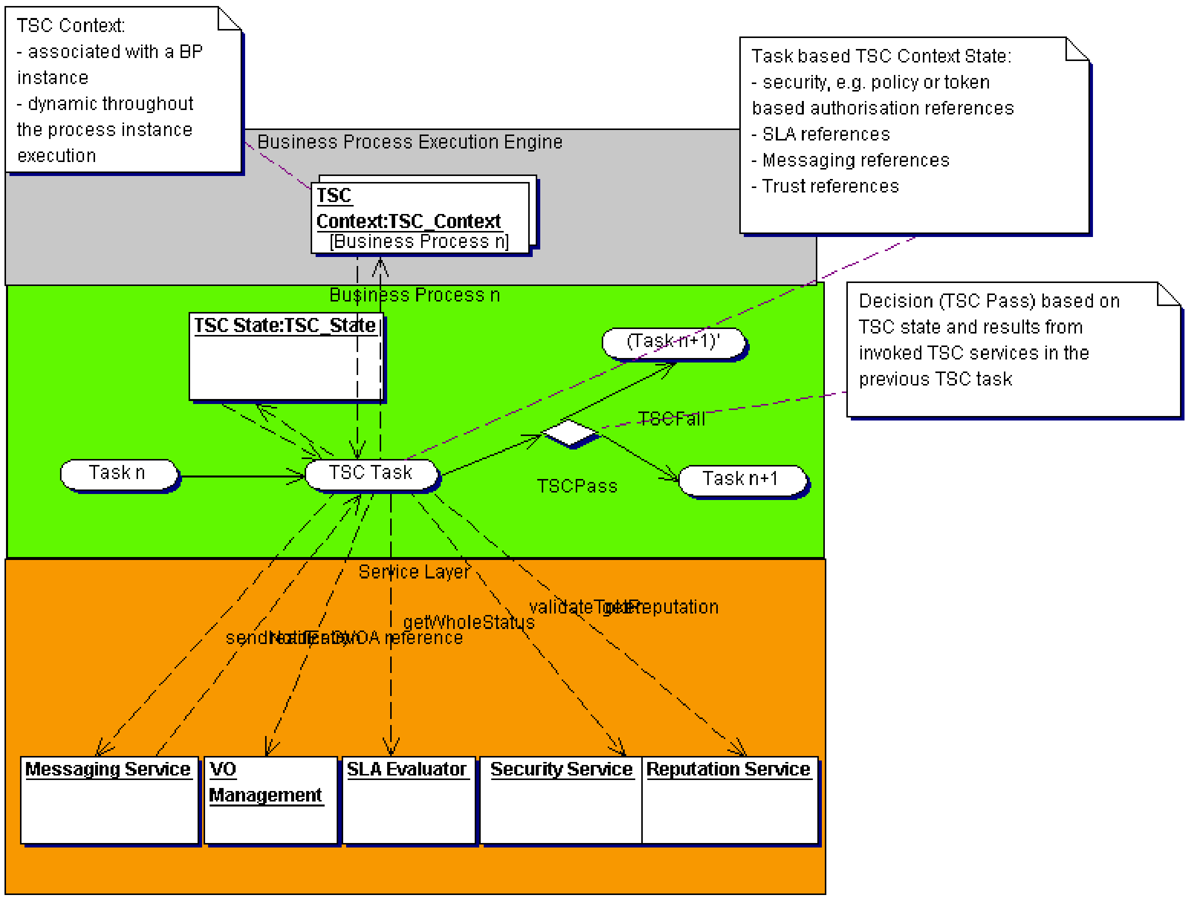
TSC Context:
- associated with a BP instance
- dynamic throughout the process instance execution
Business Process Execution Engine
TSC Context:TSC_Context [Business Process n]
Task based TSC Context State:
- security, e.g. policy or token based authorisation references
- SLA references
- Messaging references
- Trust references
Business Process n
TSC State:TSC_State
(Task n+1)'
TSCFail
Decision (TSC Pass) based on TSC state and results from invoked TSC services in the previous TSC task
Task n
TSC Task
TSCPass
Task n+1
Service Layer
validateToken
getReputation
getWholeStatus
sendNotification
getVOA reference
Messaging Service
VO Management
SLA Evaluator
Security Service
Reputation Service

Figure 7 - TSC Task

The TSC Task as shown in Figure 7 - TSC Task finally implements the TSC control within the BP. The TSC Task is already modelled in a CBP at design time, however the execution of the BP control happens during runtime, when the BP instance is executed and the TSC Task is performed.

Above activity diagram was refined taking existing input from WP2 and harmonizing subsystem entry points with current development in TrustCoM across subsystem WPs.

It is clearly visible that the TSC task brings all the previous concepts together again, only the configuration by TSC Extension Role deployment is omitted here due to the focus on runtime. The example in 3.3 provided a scenario oriented perspective on the task's function and interactions. To re-iterate in more detail:

- The TSC task is implicitly modelled in a BP at design time from a CD, but is not necessarily assigned a TSC Extension Role yet. Since a CD contains no tasks, the occurrence of an annotated TSC Extension Role denotes the requirement for a TSC task in the corresponding BP
- The TSC task receives its purpose by assignment of a TSC extension role which configures the task's attributes; this assignment can occur at design time or at runtime, prior to the task's execution
- By performing its function, a TSC task controls the BP flow or emits notifications (e.g. alerts)
- When the task is executed, control is exerted based on invoked TSC subsystem services
    - Currently, the following subsystems are entailed in the model:
        - Messaging, to retrieve or emit notifications
        - VO Management, particularly for gathering information about referenced policies from the GVOA
        - SLA, to evaluate violation of a SLA
        - Security, for token services
        - Trust/Reputation, to take a collaborating entity's reputation into account for BP control
    - The TSC task itself does not contain or operate on confidential data; service invocations are performed based on the TSC context fields assigned to the currently executed task
    - Since it is recommend to maintain the TSC context at a trusted location, the data subset which belongs to the currently executed TSC task is retrieved from the context and temporarily stored in a TSC state, only until the TSC task execution is finished.
- The "TSCPass" decision diamond denotes the BP control flow decision based on previous subsystem interaction

The TSC model is designed to integrate with other subsystem models and standards/profiles envisioned to be taken up in TrustCoM and yet open and extensible. In case of other subsystems or sources/entities having an effect on the BP control flow, those may be integrated in the model as well. This would lead to an additional TSC extension role and corresponding sections in the TSC context/state. The TSC task model would be unaffected and perform the new role as provided here.

4 Contract and Service Level Agreement Management (SICS)

4.1 The scope and focus of contract management in TrustCoM

Contracts are important enablers of virtual organisations, reducing perceived risks and making explicit the expectations of their participants. Contracts compensate for lack of trust[2] by providing guarantees that parties with low trust levels will behave properly. They can also be used to complement security, especially where preventive measures are either too costly or do not exist.

Within a TrustCoM VO, we identify two classes of contracts:

- *General VO Agreements[3] (GVOA)*: contracts that express the general rules each partner of a VO must abide to, in order to be acceptable as a member of the VO.
- *Service level agreement (SLA)*: contracts that express the specific rules that partners involved in a specific (operational) business process must abide to.

A GVOA identifies and specifies the general rules that characterise how operational business processes are to be conducted through collaboration in a VO. To be able to automate operational business processes, and to support various types of quality assurance and quality control, *each partner* must commit to provide (to other partners in the VO) certain information and offer (to other partners in the VO) certain services. On the other hand an SLA describes QoS objectives *for a specific service* as agreed by the service provider and the service consumer.

GVOAs do not accommodate the full range of features that the typical business executive expects from a contract. The many different issues covered by a typical business contract make such contracts extremely difficult to formalise and to completely handle by automated means. Hence, such formalisation and the automated handling of complete business-level contracts have to be considered out of scope and, furthermore, the added value (i.e., business profitability) of doing such formalisation is questionable. For the time being, we will have to rely on the enclosing socio-economical environment to define and evaluate such contracts[4].

[2] Buskens, V., Raub, W., Weesie, J. (2000). 'Networks and Contracting in Information Technology Transactions'. Pp. 77-81 in: Weesie, J., Raub, W. (Eds) 'The Management of Durable Relations; Theoretical Models and Empirical Studies of Households and Organizations'.

[3] The term “General VO Agreement” was chosen over the rather more standard “Collaborative Agreement” since the latter tends to cover legal, social and business-oriented aspects that are considered outside the scope of the present analysis. We focus here on a kind of “Constitution for SLAs”.

[4] For further detail on business contracts for virtual organisations see e.g. Weitzenböck, E. M. “Virtual Enterprise Model Contracts”, a publication of the ALIVE Project (Advanced Legal Issues in Virtual Enterprises, IST-2000-25459, http://www.vive-ig,net/projects/alive/).

4.1.1 Requirements

By contract management we mean management of General VO Agreements (GVOAs) and Service Level Agreements (SLAs). Unless further specified, the term “contract” covers both GVOAs and SLAs.

4.1.1.1 General Requirements

Contracts must have a formal grounding and be prone for automatic management. Therefore we shall restrict our attention to contracts taking an electronic form (i.e. electronic contracts) with formally defined guarantees to facilitate trustworthy collaboration.

The VO infrastructure should support the lifecycle of contracts. While the lifecycle of GVOAs is tied to the whole lifecycle of the VO, SLAs have in general a much shorter lifecycle. In general a VO should provide trusted services for:

- Formulation, negotiation and endorsement of contracts between VO partners.
- Identification and resolution of conflicts between the VO contracts, the security management policies of the VO constituencies and the collaborative business processes enacted.
- Secure storing of contract elements (including contract templates).
- Performance assessment (monitoring, evaluation of contracts and accounting).
- Arbitration and contract amendment.
- Termination of contracts.
- Recollection of cumulative evidence for posterior analysis.

4.1.1.2 Legal Requirements

In the legal area, TrustCoM is focusing on privacy, data protection and international issues. Consequently it is important that contracts are designed to facilitate risk analysis with respect to these legal issues.

WP9 has identified legal risk analysis as an integral part of contract negotiation. Given that TrustCoM aims at automating most aspects of contract management and that risk analysis can hardly be automated, the proposed approach is to base contract negotiation on contract templates (also called “model contracts”) that are created and analysed prior to their usage within an automatic negotiation process.

4.1.1.3 Socioeconomic Requirements

The creation of a VO and its evolution is substantially regulated by the GVOAs, and therefore these contracts need to provide the right incentives and define appropriate rules-of-the-game that facilitate sharing of services, resources and information.

4.2 Conceptual modelling of General VO Agreements (KCL)

To be able to view a set of organizations as a VO, there must be a common framework that defines what it means for these partners to collaborate within the context of this VO.

On the “business level”, the partners may need to establish a formal business relationship, which, on a legal level, defines the rights, obligations, etc. in general terms. For a VO, in the TrustCoM sense, there is a lower level of agreements – here identified as the “operational level”- that are of a different character, and that can, and should, be automated. This concerns identifying and specifying the general rules that define a virtual organization and characterise how operational business processes are to be conducted through collaboration. To be able to automate operational business processes, and to support various types of quality assurance and quality control, each partner must commit to provide (to other partners in the VO) certain information and offer (to other partners in the VO) certain services – this is described in more detail below. Without these commitments in place, a VO can only be in a germinal state. In order to operate, a VO requires the establishment of agreements satisfying a set of minimum constraints.

The distinction between the operational and legal levels of the GVOA can be reformulated as follows. A GVOA contains a main section defining the business and legal terms and conditions of the agreement, and an appendix defining the technical configuration and operational constraints of the required infrastructure, including business process instances, SLA instances and security policies to be enacted or enforced within the VO. As mentioned in section 6.3 of D15, a legal template for a VO contract has been produced by the ALIVE IST project.[5] An overview of this template is to be found in section 2.5.1.1 of Appendix B to D15. There the template is applied to the legal risk analysis of the CE scenario. In section 2 of Appendix B it is applied to the legal risk analysis of the AS scenario. Of special interest is the fact that the two scenarios do not presuppose the same contract structure.

We end this section with a number of remarks, which mainly concern the operational level.

An example of information that should be provided (by partners) is meta-data on services offered as part of their contribution to the VO. Even though this may be seen as a completely obvious requirement, it is nevertheless important to explicitly state even such things, as there must exist operational criteria that can serve as proof of mal-performance of individual partner – an important component of what we call VO Management.

[5] ALIVE IST project VE Model Contracts, Deliverable 17a (2002).

As an example of services that a partner should provide, we highlight the need for monitoring services. When a partner participates in the enactment of an operational business process, there may be critical performance or quality measures that need to be collected. Monitoring services are in place to monitor performance of dedicated tasks in the operational process and provide this data – eventually in an aggregated and already processed form – as a basis for these quality measures. As a composite business process can be enacted through decentralised distributed processing, one may end up in a situation where data need to be monitored on different sites (in different administrative domains), and where such data need to be composed or merged in order to evaluate how processing progresses. In this case, it may be necessary for monitoring to be executed on one site (belonging to one partner), and the resulting data to be used on another site (belonging to another partner), thus e.g. ensuring that required tasks have been performed appropriately.

The general need for cross-domain activities, as depicted in the two preceding paragraphs, should be seen as part of the commitments partners make when joining a VO, and such commitments may need to be valid throughout the lifetime of a VO. Hence we regard such commitments as being part of the *general VO agreement* (GVOA).

From a process-oriented perspective, the GVOA is one of the components that appear in the area of VO Management. VO management, specifically in the VO formation phase, should include specific negotiation about the GVOA, producing a GVOA that all partners commit to. If there is a higher level business oriented contract between the VO partners, this GVOA could be one of the items that the contract covers. From an operational point of view (especially in terms of automation) the GVOA controls and constrains the specific actions taken in VO collaborative work.

Just as a general contract may need to be revised if the existing contract is detected to be unsatisfactory, the GVOA could possibly be re-negotiated at different points in time. One need that must be addressed is the consequences of modifying the set of partners, or modifying role-assignment of partners in an existing VO. Hence, in the defined VO Management process for VO modification, the GVOA is one of the components that are treated according to well-defined rules.

Figure 8 provides a summarized view of the model of General VO Agreements that is emerging from the work within WP28.

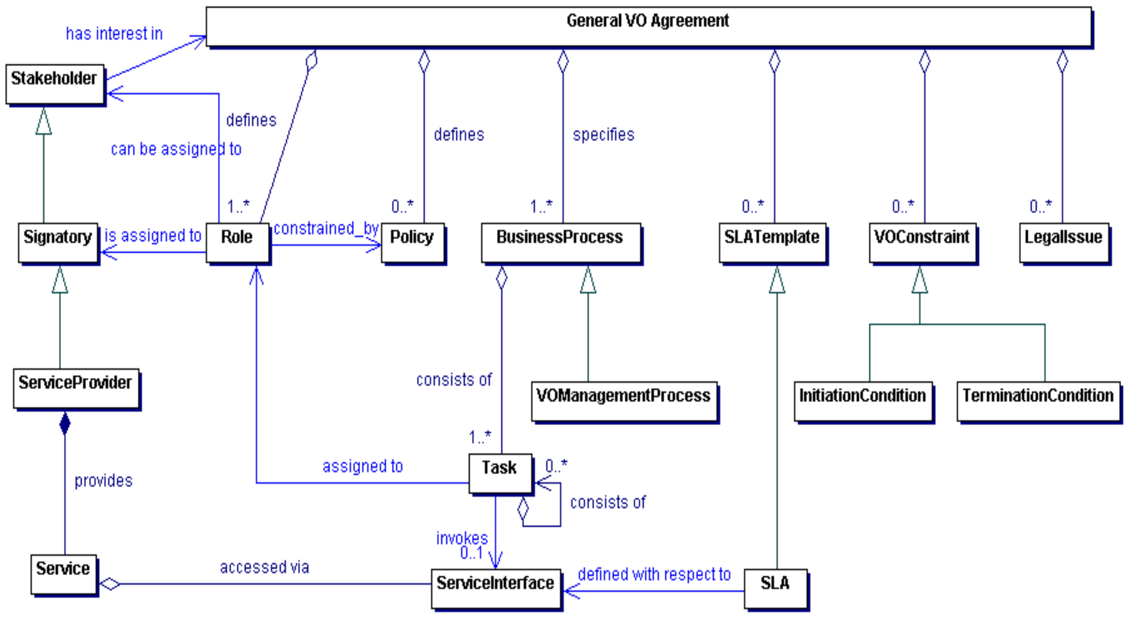

Figure 8 - Static model of General VO Agreements

This model, which has proved valuable with respect to the technological development in TrustCoM, puts legal aspects to one side. For instance, it is not said who represents the VO towards third parties (e.g. consumers) and authorities. The Alive template alluded to above assumes that it is the VO Architect. Likewise, as explained in section 6.3 of D15 “Report on legal issues”, any service provider has to comply with requirements for contracts defined in national laws based on the E-commerce Directive.

Following the template-based approach for business processes and SLAs adopted by TrustCoM, it is envisaged that the initiator of a VO chooses a GVOA template (called “community doctrine” in [6]), instantiates it and publishes its intention to build a VO on this basis. The GVOA identifies a set of VO Management processes together with a set of roles involved in the enactment of those processes. It may also list security policies and SLA templates to introduce (non-functional) constraints on roles and on the enactment of business processes. Furthermore, a GVOA may specify one or more operational business processes in accordance with the business objectives that support the creation of the VO.

When a stakeholder declares its interest in participating in the future VO, it must indicate the roles it may be willing to assume. Depending on the negotiation model, a stakeholder may also propose policy changes, trigger the deployment of new

6 Sye Loon Keoh, Emil Lupu and Morris Sloman, “PEACE: A Policy-based Establishment of Ad-hoc Communities”.

business processes, etc. Up to that point, the GVOA is considered to be non-effective.

The main goal of the negotiation process is that a sufficient number of stakeholders join the VO by becoming their signatories. When the VO Constraints of the GVOA are fulfilled, the GVOA is said to become effective.

At any time during the operation of the VO, participants can join and leave, role assignments can be altered, and VO Management processes can be modified. However, if at any point the VO Constraints cease to be satisfied, the GVOA becomes non-effective, requiring a new round of negotiation, involving membership management and related VO management processes (e.g. to replace or drop a partner) or forcing the VO into the dissolution phase.

4.3 Conceptual modelling of Service Level Agreements

At this point of the conceptual modelling work performed within TrustCoM it is important to develop models of SLAs that capture the most essential features and to avoid cramming it with unnecessary details. Since several details are related to the particular application area of the SLA, it is outside the interest of this work to describe agreements at such level. Instead, stress is put here on producing a simple model that may later be specialized with domain-application information. It is also crucial to leave architectural issues aside and concentrate on a few concepts whose realization could certainly take many different forms. One further requirement is that the model should give a basis not only for the development of SLA management services, but also for the analysis of contracts (e.g. to detect inconsistencies within a single agreement and collisions within a set of contracts).

A fundamental building block of generic agreements is the notion of *obligation*. It is worth pointing out that many concepts related to SLAs (and agreements in general) may in fact be expressed in terms of obligations. Doing so requires a model capable of expressing very generic obligations (e.g. taking into account complex performance measures) and their interdependencies (i.e. chains of obligations).

Modelling Approach

Assessing the performance of a contractual obligation implies accumulating information from different sources (monitors). Monitors are in principle independent of each other and thus define, quite naturally, a concurrent and distributed system. The same can be said of elaborate obligations that predicate on the performance of two or more services.

It is generally understood that concurrency and distribution do not lend themselves easily to formalizations where a *complete* system is modelled as a function transforming inputs into outputs. Therefore, any specification of a system of collaborating monitors and monitoring data aggregators would be strongly limited if we were to use such a formalization style. Agreement standards like WSLA and WS-Agreement do use a functional style but they partition the system into subcomponents and let each component evaluate their own set of functions. Thus, monitors aggregate data using functions (i.e. metrics), and Condition Evaluation Services (WSLA terminology) take metric values and compare them against threshold values to decide whether obligations have been violated. If the approach is kept at this simple-minded level, what we get is obligations with a single-level structure, i.e. obligations whose performance status depends only on the data produced by monitors: Monitors gather information, aggregate it functionally, and use the resulting values to evaluate a *performance predicate*. The outcome of this predicate determines whether an obligation has been violated or not. In other words, the performance predicate determines, at any time point, the current status of an obligation.

However, it is not uncommon that an obligation gets triggered by the performance (i.e. the fulfilment or violation) of another obligation. That is, obligations do sometimes depend on other, lower-level obligations. The performance of the lower-level obligation needs to be established before deciding on the performance of higher-level ones. This yields a multi-level structure for obligations. As an example, assume that obligation A is triggered by the violation of obligation B. With state-of-the-art agreement specification languages, expressing such a dependency forces one to model violation/fulfilment events (of obligation B) also as parameters to the *performance predicate* of obligation A. Moreover, since the context is fundamentally reactive (meaning that the performance of an obligation is not completely determined by the initial conditions), this approach needs to describe monitor specifications in terms of schedulers, time series and the aforementioned aggregation functions.

The resulting specification languages are unnecessarily complicated, even more when their definitions are so much tied to syntactic issues deriving from the adoption of XML. There is a need to produce a simpler conceptual model of agreements that is free from such syntactic details but that may be progressively instantiated to explain current specifications (e.g. WSLA and WS-Agreement).

Our modelling approach is based on well-studied operational models of reactive systems. In particular, we specify an agreement as a *timed event-state transition system*. The fundamental and most basic concept used in the model is that of an *event*. An event may be compared to the triggering of an alarm (e.g. when a deadline is reached, or when a monitor is to be queried), so as to inform about the violation of an obligation or a change in monitor data. In an event-state transition system, the occurrence of an event may cause a state transition, provided the current state satisfies its enabling condition.

Other researchers have proposed to model contracts with different kinds of transition systems: e.g. finite state machines[7] and Petri Nets[8]. The present model

[7] E. Solaiman, C. Molina-Jimenez and S. Shrivastava, ‘ Model Checking Correctness Properties of Electronic Contracts’. In *Proceedings of the International Conference on Service Oriented Computing* (ICSOC03), Lecture Notes in Computer Science, Volume 2910 pp. 303-318, Springer 2003.

[8] R. Zimmer, A. Daskalopulu and A. Hunter A., ‘Verifying Inter-Organisational Trade Procedures by Model Checking Synchronised Petri Nets’, *South African Computer Journal*, 1999.

attempts to bridge the gap between those formal models, usually not concerned with specifying the sources of monitoring data and/or abstracting time, and the SLA specification languages proposed for grid and web-services settings (e.g. WSLA).

4.3.1 The Abstract Model

A service level agreement defines a relationship between parties whose task consists in the delivery of one (or more) services. The purpose of the agreement goes beyond the description of functional properties for creating and identifying these services. It also covers non-functional aspects such as performance, availability and the way services may depend upon each other. The latter aspect is useful when the services co-operate to enact a business process.

This model of agreements should be understood at the “operational level” of a VO, and consequently makes no statement about the business objectives of the agreement.

As mentioned in D15 “Report on legal issues” (p. 26), further requirements for contracts and SLAs are defined e.g. in the E-commerce Directive.[9] To the extend VO members provide services, they will need to comply with the duties laid down in the directive. However, a detailed study of the latter directive falls outside the scope of the present deliverable. Legal issues related to some aspects of the directive will be addressed in further work within WP9.

[9] DIRECTIVE 2000/31/EC OF THE EUROPEAN PARLIAMENT AND OF THE COUNCIL of 8 June 2000 on certain legal aspects of information society services, in particular electronic commerce, in the Internal Market (Directive on electronic commerce), Official Journal of the European Communities L 178/1-16.

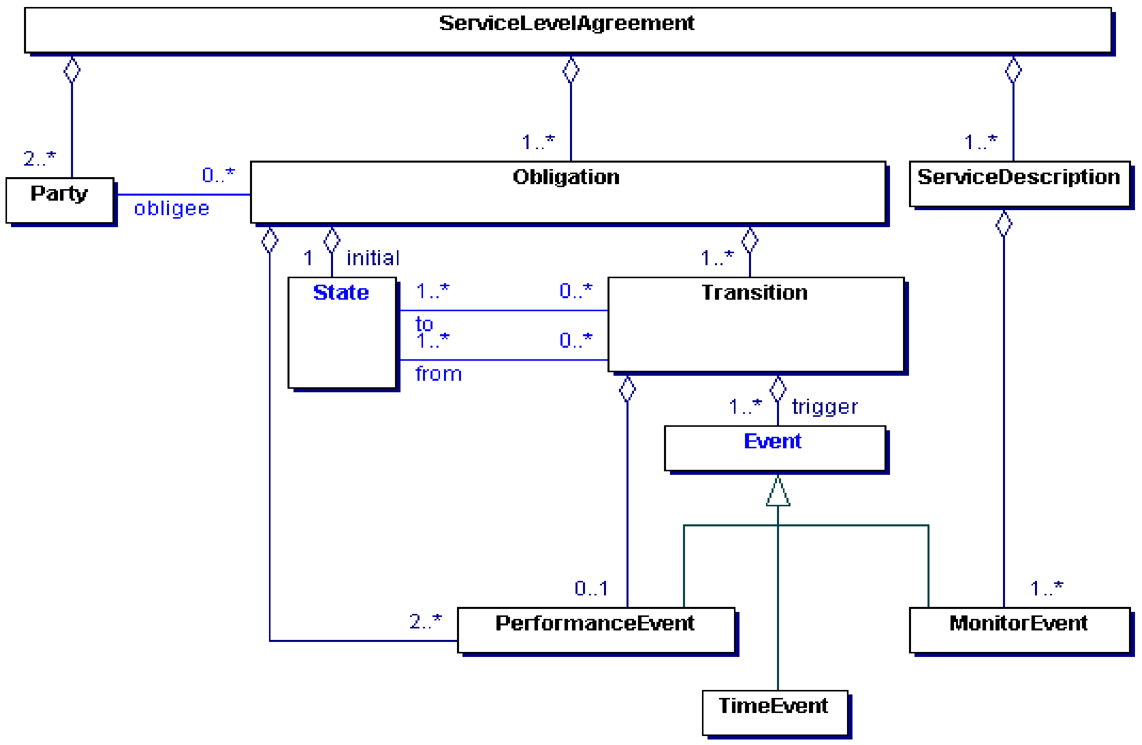

Figure 9 - Abstract model of Service Level Agreements

Figure 9 shows, in the form of a UML class diagram, the main concepts and relations in the abstract model of Service Level Agreements.

Party. A party is a VO Partner taking part in the agreement. For Service Level Agreements, the identification of parties may be indirect in the sense that the agreement may actually refer to a VO Role. In this case, the GVOA is needed to resolve the role to the identity of a unique VO Partner that is to be held accountable for the performance of the corresponding obligations in the agreement.

Obligation. A relationship that binds one party to a performance (as a payment or transfer) or non-performance. An obligation describes a behaviour using timed events and identifies the responsible party. At any given point in time the obligation is in a determinate state. The occurrence of an event may force the obligation to change into a new state. Such a change is called a (state) transition. A transition may also generate *performance events* to communicate the *violation*, the *fulfilment*, or some other information regarding the state of the obligation.

Having a responsible party does not imply that all the events mentioned in the obligation are exclusively generated by this party. Some events are generated by the components in charge of evaluation the contract (i.e. performance events), some are time related, and others may be generated by third parties (e.g. monitors when they report to the agreement evaluator or service consumers when they request services). The party holding the responsibility for an obligation is expected to see to it that the execution of the obligation never results in a violation event.

Observe that concepts featured in other models of agreement, like schedulers, triggering conditions and validity period of an obligation, are no primitive in this model. Instead, they must be represented using the more basic concept of state transitions.

Service Description. A service description is a set of documents that describes the interface to and the semantics of a service.[10] A service description contains the details of the interface (as in WSDL) and, potentially, the expected behavior of the service. This includes its data types, operations, transport protocol information, and address. It could also include categorization and other metadata to facilitate discovery and utilization.[11]

From the perspective of a service level agreement, a service description identifies the sources of monitoring events that may change the current state of its obligations. The sources of these events are naturally monitor services (i.e. services implementing an SLA Monitor interface).

State. A situation in the lifetime of an obligation (within some agreement) between two consecutive events. The state of an obligation reflects the sequence of past events that had an effect on the obligation. Events trigger transitions whose effect may change the state of the obligation.

As an example, consider the case in which a service provider is obliged to respond to a service request within 5min of the request. The event corresponding to the service request changes the state of the obligation to reflect the expectation that the service provider answers it within 5min.

Transition. The passage from one state to another. The occurrence of an event, whether it is external or internal to the agreement evaluator, triggers a state change (transition) provided that the current state and the event satisfy the enabling condition. The effect of the transition may include changing the current state of the obligation and/or emitting some performance event.

Event. A significant occurrence or happening. Events of interest for the evaluation of an obligation include: external events generated by monitors, time deadlines, performance events produced by other obligations. The latter makes it possible for an obligation to depend upon another. For example, an obligation A may become active when obligation B is fulfilled.

Monitor Event. An event produced by the services in charge of monitoring the behaviour of application services. The description of the service (see Service Description) lists the monitoring events related to the behaviour of the service.

Performance Event. An event whose occurrence is determined/identified by the agreement itself and which relates to the performance of some obligation. Both the violation and the fulfilment of an obligation may be used to trigger other obligations. The only possible cause of a performance event is the execution of a certain transition. The set of performance events associated to an obligation must contain at least two events: one to signal the fulfilment of the obligation, and one to indicate

[10] “Web Services Architecture”, http://www.w3.org/TR/ws-arch.

[11] Ibid.

its violation. In the next section, we partition the state of an obligation and put each partition under the control of a separate “enforcement component”. In such a context, performance events may as well be used to communicate information between enforcement components (see Section 4.3.2). A common use of these “internal” performance events is to model the communication of aggregated monitoring data between components within the SLA Management subsystem.

Time Event. An event reflecting the passage of time. This class is a conceptual artifact, not expected to have a corresponding physical existence in the architectural models.

Note that the concept of (regular) time intervals is not included in the model, since they can be easily represented by the pair of time events that mark their beginning and end points.

4.3.2 A Concrete Model

The generality of the abstract model presented in the previous section would be of little use if the model could not be refined (i.e. instantiated) into more concrete models. Given that the TrustCoM Framework has at this point adopted WSLA (Web Services Level Agreement) as the specification XML-based language for SLAs, this section refines the abstract model to represent the main concepts that are used by the WSLA Profile for TrustCoM (see D18 TrustCoM Framework).

Figure 10 illustrates the approach, proposing a refinement of the classes Event and State appearing in Figure 9. The main idea is to partition the set of events to distinguish between basic measurement data, SLA parameters and violation notifications. Basic measurement data is raw data obtained from Basic Monitors which in most cases will be implemented using some sort of service instrumentation (cf. D9 TrustCoM Architecture). SLA Parameters are computed using metric functions from basic measurement data and other SLA Parameters; and violation notifications follow from the evaluation of Service Level Objectives (SLO), called obligations in the previous section. The components responsible for generating each of these sets of events are --according to the TrustCoM Architecture-- the Basic Monitor, the Aggregating Monitor and the Evaluator, respectively. Each of these components controls a part of the state associated to the obligation in the abstract model of the previous section. Observe that SLA parameters are communicated among Aggregating Monitors and Evaluators using (internal) performance events. Therefore, the communication of an SLA parameter is associated to the event of class SLAParameter, which refines class PerformanceEvent.

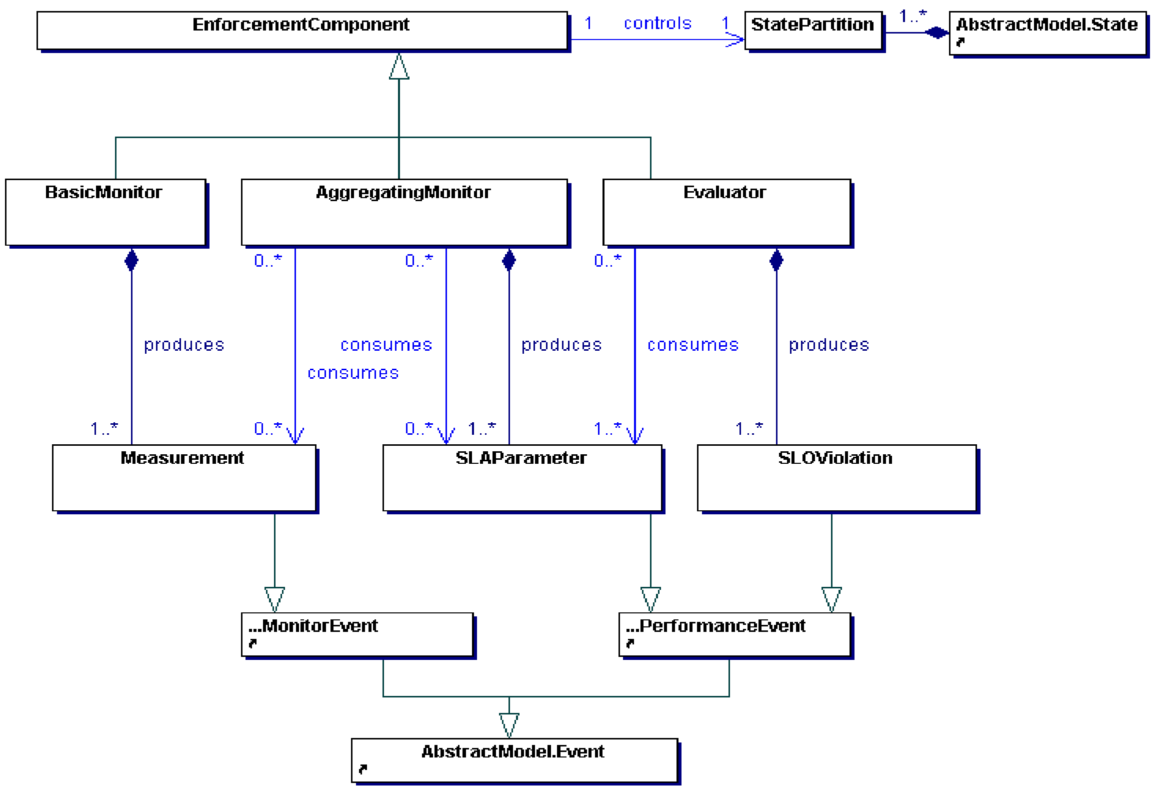

Figure 10 - Refined model of states and events

The concrete model constrains the shape of transitions too. Each transition is assigned to exactly one of the three enforcement component types. Furthermore, each transition is made to depend on and modify only the state partition that is under the control of its assigned enforcement component. For example, the state transitions associated with an Aggregating Monitor depend on and modify only the state partition that is under the control of the Aggregating Monitor. In this way, the state is distributed into all the enforcement components and each set of transitions assigned to a component fully defines the component's behaviour.

Finally, enforcement components communicate by emitting (i.e. producing) and receiving (i.e. consuming) events. For example, an evaluator component's transitions get triggered by SLAParameter events and may emit SLAViolation events.

4.4 Negotiation of Service Level Agreements (ATOS)

The participants of a negotiation are the Service Provider (SP), who offers its services and the Service Consumer (SC) or someone acting on its behalf.

For version 1 of the framework, the negotiation protocol is restricted to a single round where the offer, made by the service consumer based on the SLA template, is either accepted or rejected on the spot by the service provider.

One of the possible purposes of the SP is to provide several services to the SC. When a SP has deployed a service and wants to publish it into a TrustCoM Virtual Organization (VO) it is also necessary to provide at least one SLA template document that will be the base of a future contract between the SP and the SC.

The main goal is to reach an agreement that will establish the conditions of usage for a concrete service. The negotiation phase will terminate when all the services have been contracted and all signed contracts have been saved into a repository.

In the negotiation phase, the SC, in agreement with the conditions that offers the SP, decides the level of QoS required. The participation of the Policy Subsystem and a service in charge of managing the execution and management of service instances (e.g. EMS) provided by the Enterprise Network Infrastructure is also necessary to ensure the availability of the requested resources. Afterwards, if the negotiation has been successful, a contract is made between the SC and the SP for guaranteeing the achieved agreement. It will be stored in the SLA Repository.

4.4.1 Actors

The following actors have been identified with some participation with any of the entities that make up the negotiation process at the SLA Management subsystem.

Service Consumer: typically the end user that consumes the services provided by the Service Provider

VO Manager: Its role is to be in charge of managing the negotiation for all the services that are demanded by the consumer in order to concrete when and which will be the service providers that will host the service execution. Normally acts as a broker between the consumer and providers. It will manage the creation of the VO with the resultant set of service providers.

Policy Subsystem: SLA Negotiator also interacts with Policy Subsystem in the negotiation phase, when is necessary to know the metrics that define the mapping from high to low level parameters. These policies are defined at application level and are stored in a policy database.

EMS (Execution Management Service)[12]: it will provide information about service availability. It is responsible for checking the service state and establishing a pre-reservation of resources for a given period.

SLA Negotiator: A component providing support for negotiating agreements. The users of this component will include VO management processes in charge of signing agreements with service providers. A negotiator only offers functions and protocol implementations. The actual logic determining what is to be considered a successful negotiation lies outside the SLA Management subsystem.

SLA Signer: This local component implements one of the sides in the signing protocol chosen for the VO and admitted by the Notary. Different signing protocols would then require the instantiation of different SLASigner components.

Notary: This (trusted third party) component witnesses the signing of SLAs between providers and consumers

SLA Manager: This component functions as a coordinator for the different components in the SLA Management subsystem. In particular, it is responsible for the configuration of monitors and evaluators. It associates the Monitors and SLA Evaluators with an SLA and connects them with each other through the Notification subsystem. It can also be called by the VO Manager.

4.4.2 Negotiation Phases

The negotiation can be split in two phases:

- Negotiation of contract
- Establishment of contract

This division will facilitate the negotiation of an aggregated service provision (set of services that are composed and enacted 'on the fly' in response to a user's requirements), as there will be no agreement for the aggregated service until all contracts from the services composing it have been accepted.

[12] OGSA https://forge.gridforum.org/projects/ogsa-wg/document/draft-ggf-ogsa-spec/en/23

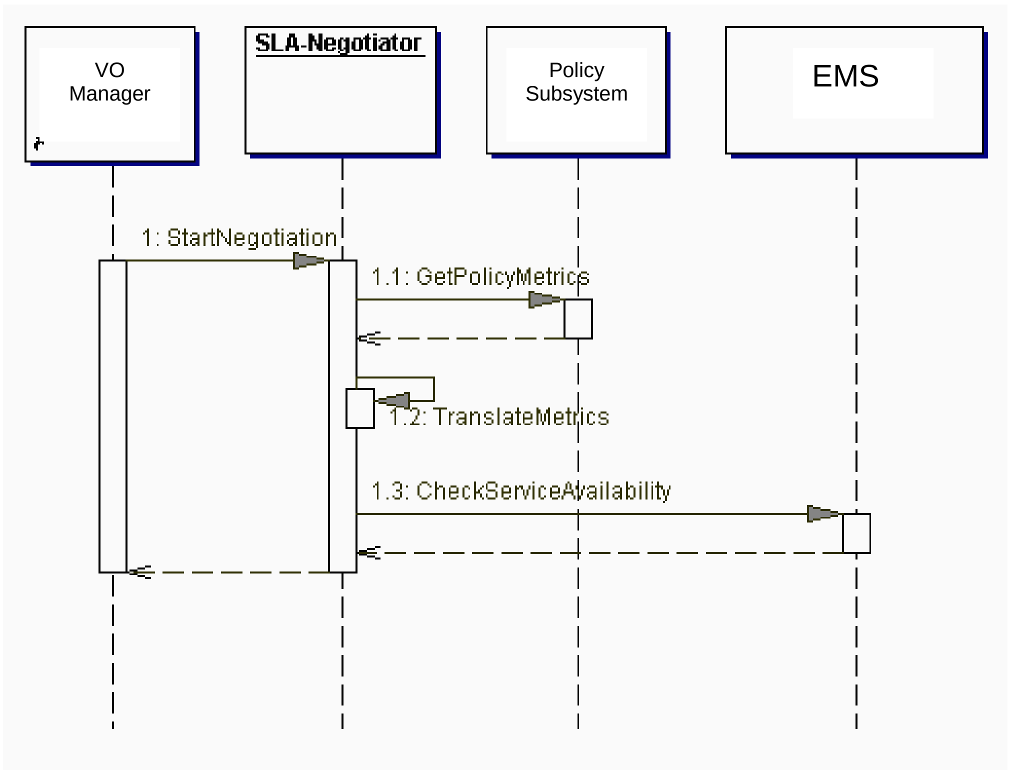

Figure 11 – SLA Negotiation Process

4.4.2.1 Negotiation of a contract

4.4.2.1.1 Start Negotiation

The VO Manager acts on behalf of SC and determinates values for the QoS that it is asked for. After it starts a negotiation to check whether the SP is able to fulfil what is required. The SLA-Negotiator receives the parameters from the VO Manager and now is able to check the availability of the resources that the Service Provider offer. There are two main tasks in this use case:

- The Service Consumer sets the application input parameters, which determine the quality of service, selecting them from the list of parameters proposed by the Service Provider
- These parameters are sent to the VO Manager that starts the negotiation process with SLA-Negotiator.

4.4.2.1.2 Mapping business metrics

The SLA-Negotiator queries the Policy Subsystem for mapping the negotiation policies from high (business) to low (measurable metrics) level parameters. The flow of events is the following:

- The SLA-Negotiator retrieves the metric policies from the Policy Subsystem
- The SLA-Negotiator translates the metrics received into QoS parameters understandable from the EMS

4.4.2.1.3 Service Availability

The SLA-Negotiator will query the EMS provided by the EN/VO in order to check the availability of services and resources at the Service provider in a given period of time (for its future execution) according to the QoS parameters. The flow of tasks to be performed is:

- At a SLA-Negotiator request, the EMS checks the availability of the Service Provider to offer a service with this QoS
- The EMS returns to the SLA-Negotiator the result of this availability.

4.4.2.1.4 Confirmation of QoS

- The SLA-Negotiator accesses the SLA template and sets the QoS section. This template will become a contract if the agreement is confirmed.
- The SLA-Negotiator returns the availability of the chosen SP to the VO Manager. There are two alternatives:
- Service is available: (the Manageability service has done a pre-reservation of the appropriate resources, only until contract is created). The next step is to confirm the reservation.
- Service is not available: The negotiation process should start again. The Service Consumer will be able to change the QoS parameters or to search for another Service Provider.

4.4.2.1.5 Confirm reservation

The EMS is notified to confirm the pre- reservation of a service usage of a concrete service during the negotiation phase.

Flow of events:

- VO Manager communicates to SLA-Negotiator that the agreement has been reached and confirmed.
- SLA-Negotiator notifies the EMS that the pre-reservation must be now a reservation.

4.4.2.2 Contract Establishment

After the reservation is confirmed, an SLA-Contract is created. It will be the base of the fulfilment of the required QoS. This contract will be stored in a repository.

Flow of events:

- The agreed SLA is signed by Service Providers and Service Consumers
- The contract is saved in a trusted third party repository (Notary)
- The SLA document is also stored in a SLA repository provided by the Enterprise Network framework for the execution phase.
- The SLA Manager is also notified of this registration. At execution phase it will send the SLA to the operating Monitors and Evaluators modules.
- The SLA Manager notifies the VO Manager of the readiness for operation.

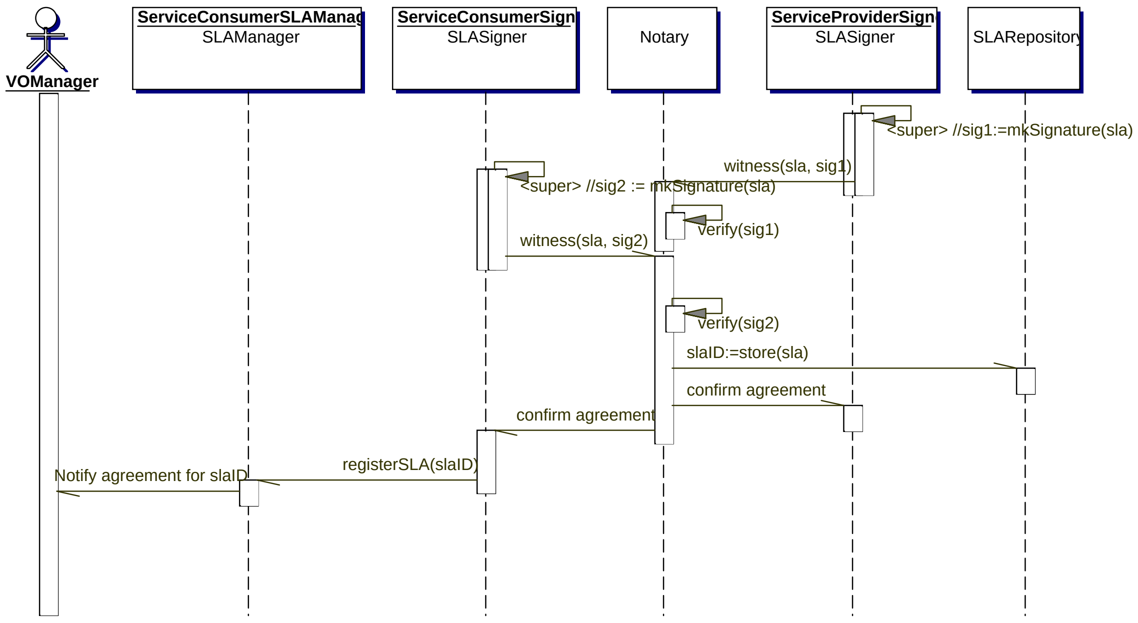

Figure 12 – Contract Negotiation Process

5 Trust and Security Management (ETH)

5.1 Introduction

As a result of the workpackage restructuring, trust and security management does not cover the whole are of security-related topics anymore instead, trust and security focuses on the establishment and maintenance of trust relationships with a priori unknown partners from foreign authentication and authorisation domains.

Establishment of trust relations covers topics like credential-based authentication and negotiation, whereas maintenance of trust relations covers topics like management of reputation.

Other security topics, access control policies being the most prominent example, are dealt with in the context of other topics. For example, access control policies are handled under the general topic of policy management.

5.2 Reputation Management

Members in the VO will need to carry out reputation management, e.g. qualifiying their trust relationships with other entities in the VO. This information can be used to learn more about the behaviours of the entities in the VO and to make trust decisions about existing and new entities in the VO.

The following diagram shows the core conceptual model of reputation management.

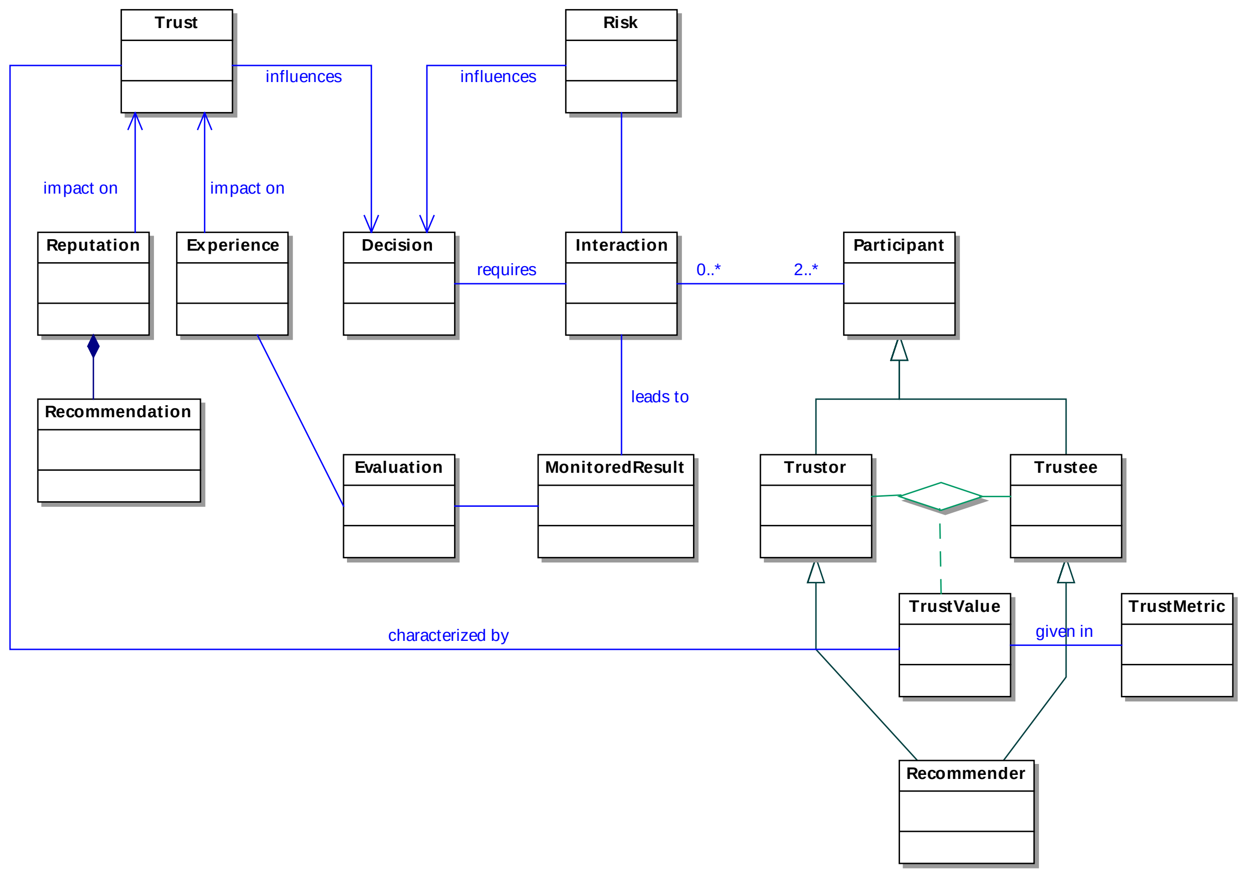

Figure 13 - Conceptual Model for Reputation Management

When a VO member needs to make a decision (for example provide access to his resources) he may need to look both at the trustworthiness (or reputation) of the entity and the risk of the action. This leads to the two central concepts in Reputation Management, which are explored in detail below: **Trust** and **Risk**

Trust is especially important to establish when there are two factors present in an exchange:

- Uncertainty, which entails an element of risk, and
- incomplete information, the information asymmetry problem, which may give rise to opportunistic behaviour.

We have an intuitive understanding of the notion of trust because we experience and rely on it on a daily basis, however a definitive definition is difficult as there are many different manifestations of trust. Two common definitions of trust consider different aspects of the notion: **Reliability Trust** and **Decision Trust**.

Reliability trust can be defined as: trust is the subjective probability by which an individual, A, expects that another individual, B, performs a given action on which its welfare depends. This definition includes the concept of `reliability' as a probabilistic relationship between the trusting and trusted entities. Extending the concept further, trust in an entity involves an assessment of the consequences of failure to perform the action that the entity is being trusted to do, the element of risk.

So trust should be placed within the context of risk, even if the probability of failure is low the risk of failure may be such that trust can not be established. Incorporating this notion of risk we can define **Decision Trust** as: trust to the extent to which one party is willing to depend on something of somebody in a given situation with a feeling of relative security, even though negative consequences are possible. This definition is broader and includes concepts of 'dependency' and 'attitude' towards risk.

These definitions of trust implicitly incorporate context and personal attitude, so are dependent on the domain and the individual's attitude to risk a unique trust model would evolve. The individualised perspective inherent in trust models produces a subjective view dependent on both the unique relationship between trusting and trusted entities and the context of the transaction. This dependency on context is in line with our aim of developing a generic trust and reputation architecture that is capable of representing different trust models and metrics.

Trust can be determined based on experience and/or experience of other parties and accuracy of this information. Therefore, two sets of information can effect the evaluation of trust relationships:

Direct Trust Experience: These are based on trustor’s direct interactions with the trustee. Experience can be both positive and negative. A Monitor will need to provide feedback on the outcome of the interaction (experience). The interaction may violate trust relationships, which can be a result of many abuses such as:

- Consuming more resources than requested
- Leaving behind data and not doing “garbage collection” after using the resources
- Going to places out of the allocated boundary
- Instantiating tasks they are not supposed to instantiate

Recommendation & Reputation (Transitive Trust): These are based on the opinion of third parties computed using their direct experience. A **Recommendation** is based on single opinion of another entity. The word “Recommendation” may suggest a positive opinion, but the model also supports negative recommendations. When there are multiple recommendations on an entity, then these recommendations will be combined together to form **Reputation**.

When assessing each of the recommendations, one needs to consider trust in the recommender. One way to handle this would be to, assign weights to recommenders according to their trust value. The result of the experience could be used to update the trust in the recommender i.e. what the recommender is trusted to recommend. **Recommender trust** will be based on consistency of recommender’s previous recommendations.

Reputation Trust: Reputation implies a generalisation or aggregation of individuals' perspectives on a given entity. The implication is that some social network has a collective view about an individual or group. Reputation is defined as `the beliefs or opinions that are generally held about someone or something'. This definition reflects the view, that trust is a more individual, personal experience whilst reputation has some collective perspective typically formed by others and utilised by an individual if they do not have personal experience of the entity under consideration. This definition also implies some longitudinal study: Beliefs and opinions are formed for a reputation over time. Thus many systems model reputation as an evolving characteristic and do so by calculating reputation as some aggregation of trust from the community of entities over which the reputation to an entity is being measured. As reputation can be a group attribute gaining membership of a group can allow an individual to automatically inherit the group's reputation, if a group is seen as reputable then members will automatically be perceived in the same way. Reputation is therefore an objective measure resulting from the aggregation of many evaluations. These aggregated evaluations can be collected either by some centralised authority as in the case of eBay or Amazon or by employing a decentralised approach by pro-actively requesting evaluations from other entities who have interacted with the entity under evaluation.

Risk: There are two alternative views of the relationship between trust and risk:

- **Risk Driving Trust**: In this view the level of risk determines the necessary level of required trustworthiness, i.e. risk drives the decision making. In this context the trusting decision we have to make can be expressed as: for a particular context c, which entails a level of risk r, how trustworthy should the principal p be in order to be allowed to enter the context c?
- **Trust Driving Risk**: The aim of this view is to protect ourselves by only collaborating with principals that are likely to behave and as a result an interaction with them is not very risky. In this context the trusting decision we have to make can be expressed as: for a particular context c involving a particular principal p, how much risk are we willing to accept by allowing principal p to enter c ?

VOs seem to fit better with the latter view. The former view seems more natural in a safety critical systems setting, where one cannot allow to exceed a certain level of risk. In more business-oriented setting, like in TrustCoM, where risk can be mitigated by other means as well, for example by legal contracts, the latter view seems more appropriate.

If we assume Risk is represented as:

Risk Exposure = Loss x Probability of Occurrence

Often we will know the loss but not the Probability of Occurrence. The Probability of Occurrence will depend on the trustworthiness of the entity, as well as other factors outside trust such as incidents/threats, recovery, frequency of the events, and etc. Therefore, the second approach Trust Driving Risk is not quite true, too.

Decisions such as access control, role assignment, choosing a new member, or removing entities from the VO can be based on trust and/or risk. Rather than treating them as one predicate, it may be more sensible to have separate predicates for trust and risk. How one influence the other and how they will be used can be defined later as part of the policies.

5.3 Security Token Services and Trust Negotiation

The first issue that is covered here, is how to issue and validate **Security Tokens** that are valid for cross-partner operations. In Trustcom, different **Security Token Services (STS)** on different levels are combined for this purpose. The basic operations model that is supported by this model is that a web service or client inside a VO partner requests a security token from its own STS, and uses this security token to invoke a web service that is operated by another partner. Other operational models are possible, too. For example, the client invoking the web-service directly, and the web-service requests an appropriate token from the client's STS. However, this model would potentially be vulnerable to DoS attacks.

A **Top-Level STS** is responsible for controlling all existing collaborations for an organization (e.g. a company). An 'existing collaboration' in this context is the fact that the organization is a participant (partner) inside a VO. A top-level STS can control participation in VOs by either

- issuing claims itself directly about the VO participation details, or by
- issuing a delegation claim that authorizes a **Mid-Level STS** inside the organization to issue claims about the VO participation details.

Additionally, a **VO-Level STS** is responsible for controlling the participation of the organisation itself in the VO, again by issuing appropriate delegation claims to the top-level STS of the respective organisations.

An overview over this structure is depicted by the following diagram:

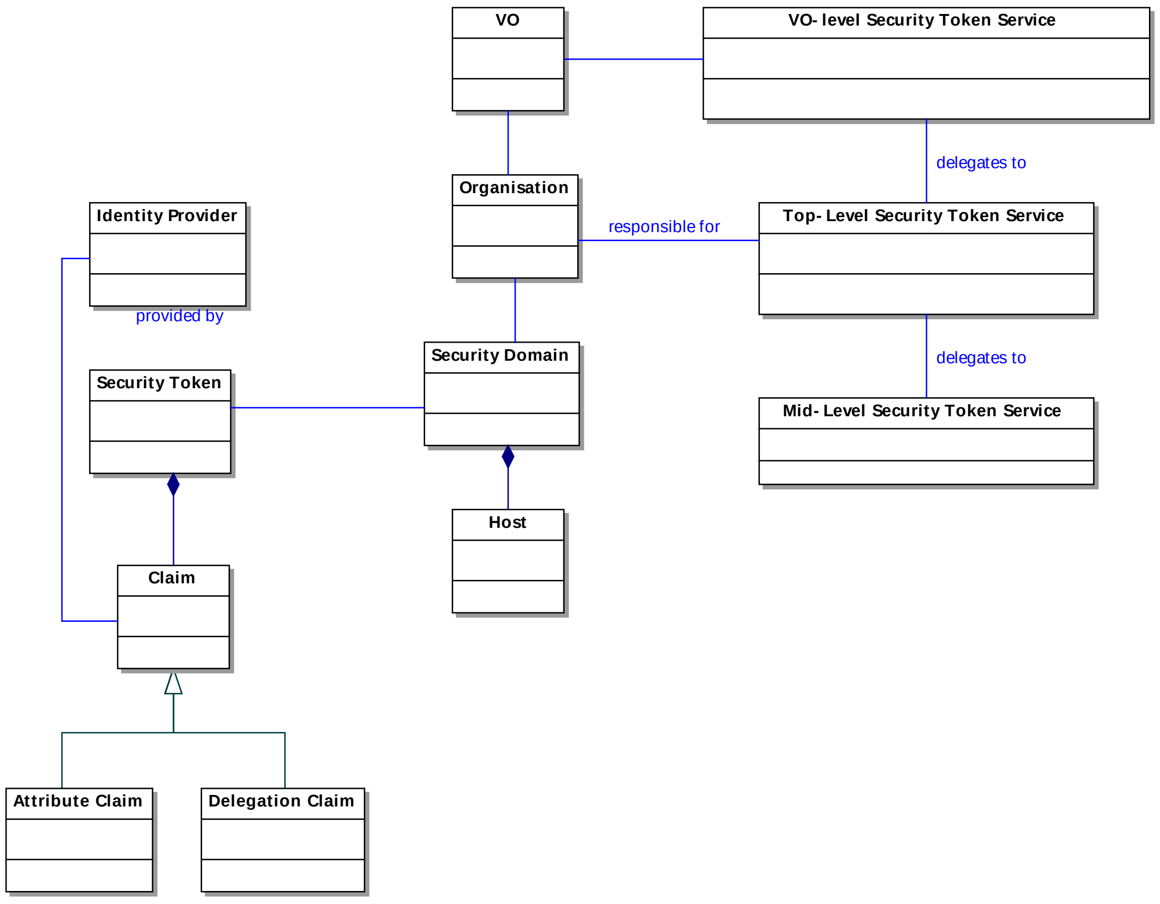

Figure 14 - Conceptual model for Security Token Services

5.4 Trust Negotiation

Trust Negotiation (TN) covers the topic of exchanging disclosure policies and claims (i.e. security tokens) in order to establish an initial trust relationship between two mutually unknown parties. Trust negotiation in Trustcom is used as a mean to “bootstrap” a trust relation between the VO Initiator (or the VO Manager) and a potential VO partner when no reputation information about the potential VO partner is available. This applies mainly to identification/formation phase of the VO life cycle as well as to the evolution phase when the need might arise to find and associate a new partner to the VO, after the expulsion of a former VO partner.

The mutual exchange above described is needed since each party might require “guarantees” to the other party before releasing its own credentials or claims.

The final result of a successful trust negotiation (initiated for example by the VO potential partner, previously invited by the VO Initiator to a trust negotiation) consists on the release to the potential VO partner of a Security Token, issued by the Security Token Service belonging to the VO Initiator domain.

o Conceptual Model Overview

Each **Party** in a **Trust Negotiation** owns a set of credentials, which are usually issued by Certification Authorities (CA), and a set of claims issued by the party itself or by a STS, if present. Altogether, credentials and claims describe attributes characterizing the owner, and they are used as a means to certify properties of the parties.

With respect to the trust negotiation process, the party that initiates the negotiation is the Client, while the other party is the Server.

Disclosure Policies regulate the release of credentials and claims to other parties. Each party involved in the negotiation process has its own disclosure policy, that state the conditions under which a credential owned by the party can be released to the other party during the negotiation process. Disclosure policies are exchanged to inform the other party of the trust requirements that need to be satisfied to advance the state of the negotiation.

A disclosure policy consists of a set of rules.

Each rule specify the condition under which a credential can be released (disclosed) to the other negotiating party, or if the credential can be released unconditionally.

In the former case, a rule has the following form:

- $CT \leftarrow P_1C_1), P_2(C_2), \ldots., P_n(C_n)$

where:

- CT is a credential/claim type;
- each P_i is a credential/claim type and each C_i is a condition defined over the attributes of P_i.

and CT can be released only if all the $P_iC_i)$ are satisfied.

In the latter case the rule has the following form:

- $CT \leftarrow$ DELIV

For a given credential, multiple conditions can be defined. In such a case, the conditions are considered as alternatives (and are considered both applicable).

Disclosure rules are distinguished in Pre-requisite rules and Requisite rules. The former MUST be satisfied before the trust negotiation can continue. Basically, they are the minimal set of “properties” that a party must demonstrate to possess to the other party.

While a disclosure policy states under which conditions a credential can be released, a negotiation strategy governs the moment when the credential themselves are released and verified. Indeed, credentials can be released after that both parties have verified that their disclosure rules are mutually acceptable; one drawback of this negotiation strategy is that during the policy evaluation phase, privacy can be compromised since there are no guarantee about other partys' honesty, until actual disclosure of the credentials. A party can use a disclosure rule received by the other party to determine the value of sensitive attributes without the credential ever being disclosed.

A trust negotiation is organized in **phases**.

- Introductory phase
- Policy evaluation phase
- Certificate exchange phase

The introductory phase begins when a client contacts a server asking for a resource R. In this phase, the information needed to satisfy the Pre-requisite rules of the Disclosure Policy of the parties are exchanged.

Policy evaluation phase begins when the previous phase ends successfully.

During Policy evaluation phase, both client and server communicate each other their own disclosure rules adopted for the involved resources. The goal of this phase is to determine a sequence of client and server certificates that when disclosed enable the release of the requested resource, in accordance to the disclosure rules of both parties.

If this phase ends successfully, one or more trust sequence are determined, each one consisting in a sequence of credentials. Once the parties choose the trust sequence to be used, the exchange of the certificates constituting the trust sequence begins. Each party discloses its certificates in the order defined by the trust sequence. Upon receiving a certificate, a party verifies that its own disclosure rules associated to the credential are satisfied, checks for certificate revocation, validity date and authenticates the ownership (for credentials).

The conceptual model of this negotiation process is depicted in the following diagram. For operational details, we refer to the corresponding description in the TrustCoM Reference Architecture.

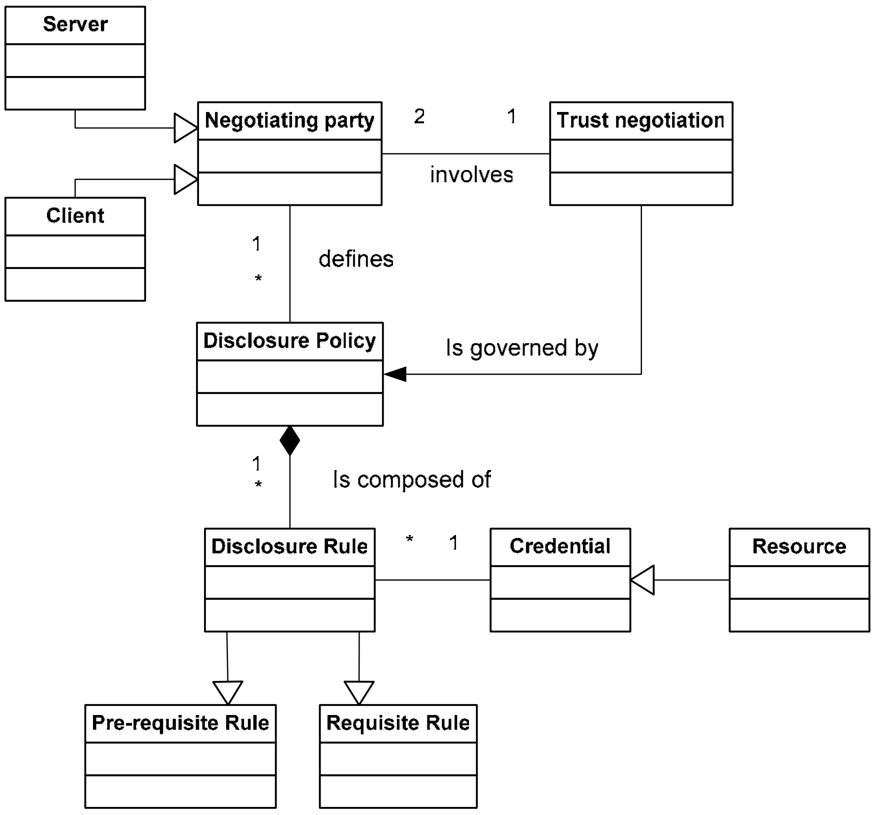

Figure 15 - Conceptual Model for Trust Negotiation

6 Representation, Deployment and Monitoring of Policies (ICSTM)

Policies define choices in the behaviour of systems. Beyond this basic definition there is little consensus in the research community on what policies are or how they should be represented. Interpretations of the *policy* concept range from basic parameter values to assertions (WS-Policy), event-condition-action rules (Ponder, PDL, TMF, ...), goals, utility functions or deontic logic specifications. In TrustCoM *policy* is also used with different meanings but each use is restricted to a specific sub-system and context in the project. We will briefly enumerate the different uses of the term and then focus on one of the components. Trustcom includes policies in the form of:

- Configuration assertions - as defined in WS-SecurityPolicy and WS-Trust which relate to the configuration of web-services and issuance of tokens. These policies are described in more detail in the Trust and Security section of the Conceptual Model.
- Monitored conditions - sometimes also referred to as Obligation policies which specify the obligations of a client or provider and are specified as part of a Service Level Agreement (SLA). These policies are described in more detail in the SLA section.
- Access Control Policies - include both authorisation policies and delegation policies and are specify in order to grant permissions to access services or to issue permissions.
- Credential Disclosure Policies - regulate the release of credentials to other parties. The specify the conditions under which a credential can be released to other parties. The mechanisms for enforcing these policies are defined and implemented in the Trust and Security Workpackage. Their deployment may be integrated with the deployment of the access control and adaptation policies in one of the subsequent phases of the project.
- Adaptation policies - also sometimes referred to as Obligation policies are in essence event-condition-action rules that determine how components should adapt in response to events arising as part of a VO. Events may include the arrival of a new participant or departure of an existing one, failures, access control violations, changes of trust or reputation that exceed given thresholds.

In the reminder of this section we will concentrate on Access Control policies and Adaptation Policies as their realisation will be part of the Policy Subsystem.

o Conceptual Model Overview

The following diagram summarises the relationship between the policies considered in this sub-system and the other VO concepts. The General Virtual Organisation Agreement (*GVOA*) represents the specification of the VO in operation. Here we consider its representation as a set of specifications related to the behaviour of participants in the VO regardless of which parts of the specification were pre-defined and which ones were negotiated by the participants during the formation phase. The GVOA defines the *roles* that the participants play in the VO as well as the agreed Service Level Agreements (*SLA)* that govern the interactions between *services*. Roles are assigned to *Tasks* as defined in the Business Process which include performing operations on *services*. Services are in turn provided by participants in the VO and must therefore be associated with the *role* that provides them. *Policies* in essence define the relationships between subjects and targets. Subjects are any entities that initiate invocations within the system by virtue of their membership in different roles whilst targets are the services on which those invocations are performed.

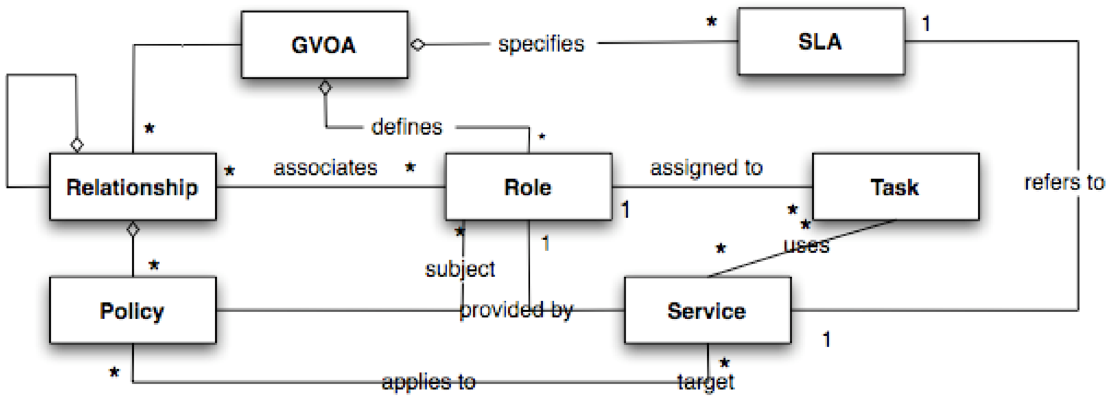

Figure 16 - Policy Relationships to the other VO Concepts.

There can be numerous policies within a VO including both adaptation and access control policies. As a consequence it is necessary to be able to group them in order to be able to define units of encapsulation, aggregation and deployment within the VO. *Relationships* between roles group the policies that govern interactions between those roles at the VO level. For example, in the CE scenario several relationships can be defined between the aerospace organisation and the analysis and design providers, storage providers etc. These relationships corresponding to storage provision, provision of analysis services etc. comprise the policies which apply to those roles. Note that *relationships* can be instantiated dynamically and their instantiation can be triggered by existing policies. Thus, for example, when a new storage provider joins the VO, new relationships can be instantiated to govern the interactions between the new member and the existing members of the VO. Relationships can also be nested thus allowing to define scopes for the policies they group.

We shall now examine in more detail the different types of policies and their relationships to the other elements in the VO (see Figure below). In essence, two types of policies are of interest: Obligation policies in the form of event-condition-action rule, access control policies. Access Control policies can be either:

- *Authorisation* policies, which define which operations *subjects* should (or should not) be able to perform on target services.

- *Proxy-type Delegation* policies, which permit subjects to grant privileges which they possess to grantees to perform an action on their behalf.
- *Administration-type Delegation* policies, which permit subjects to grant privileges to *grantees* according to their administrative authorities.

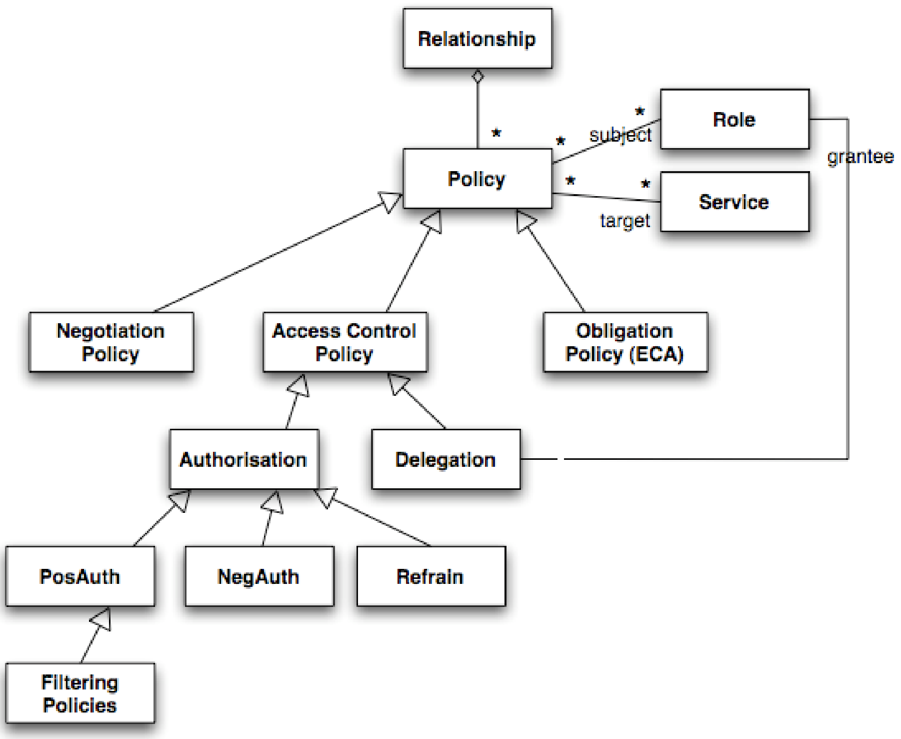

Figure 17 - Policies in the policy-subsystem

Authorisation policies are further sub-divided between positive authorisations, which permit actions to be performed, negative authorisations which prohibit actions from being performed and refrain policies which specify actions that subjects should refrain from performing. The latter are different from negative authorisation policies in that they must be enforced at the subject rather than at the target. *Filtering* policies are a specific type of authorisation policies which permit to transform input and output parameters of an action. For example, a location service might only give internal users access to detailed information such as the room a person is in, while external users would only see whether a person is at work or not. These are a form of authorisation policy in which an operation is performed and then a decision made on how to return the results, rather than making a decision on whether to permit the operation.

Both *refrain* policies and *filtering* policies are aimed at case scenarios where it is necessary to deal with information disclosure. These require additional mechanisms which may not become available during the lifetime of the TrustCoM project and may also impair interoperability. Therefore, most of the work will focus on obligation and authorisation policies.

o Obligation Policies

Obligation policies specify actions that must be performed when certain events occur. For example, security management policies may specify that the security configuration of a web-service needs to be changed in response to policy violations and intrusions. Obligation policies are enforced by dedicated agents and specify (see Figure 17 below):

- The *events* that trigger the policy.
- The *operations* and *services* on which the operations may be performed.
- *Constraints* which limit the applicability of the policy. Constraints are boolean expressions which are specified in terms of either context values or attributes received as part of the events that trigger the policy.

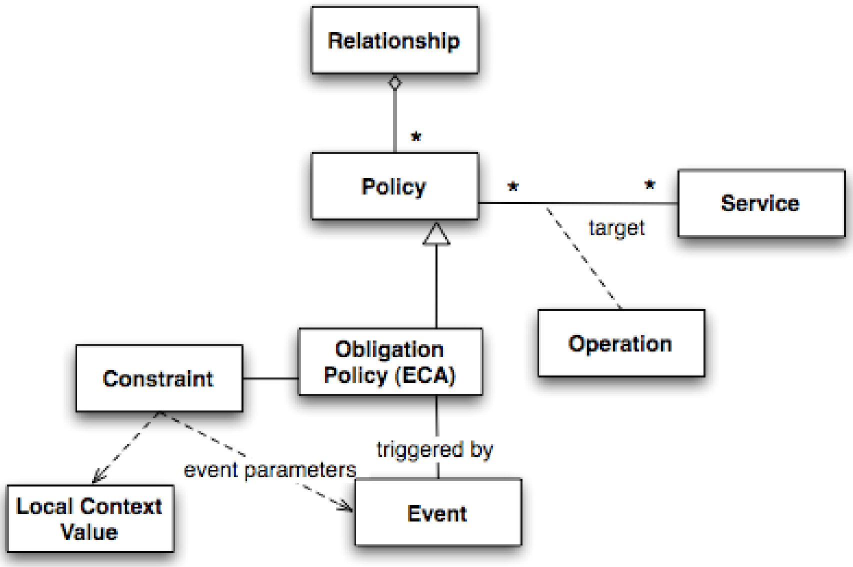

Figure 18 - Structure of Obligation Policies

Note that *operations* are here considered to be generic and can be any management operations that the service supports. Where possible, operations may load new configurations into the target web-service in the form of WS-SecurityPolicy assertions.

Within the TrustCoM framework obligations in the form of condition action rules will be used for varied purposes. For example, policies can be used in order to trigger re-configuration or redress actions when SLAs are violated, trigger reconfiguration of the security configuration of the web-services when intrusions are detected or particular levels of risk are associated with the transaction and trigger VO membership procedures when the reputation or trust in participants falls below a minimum required threshold.

Note that a second type of "obligation" policies which corresponds to the deontic concept of obligation is used within the TrustCoM project. This type of policies are the main components in the SLAs where violations to the agreed Quality of Service may occur. This means that the policies cannot always be enforced but their violations need to be detected and recovered.

o Access Control Policies

Authorisation policies correspond to the access rights that a partner needs to have to information and resources on other partners in order to be able to execute its assigned tasks. For example, if a manufacturer has the task of building a particular component within a collaborative engineering process, it needs to have access to the appropriate design specifications and test data. The overall structure of authorisation and delegation policies is represented in Figure 18 below. Each policy identifies the subjects (i.e., entities which perform the invocations) the target web-services on which operations are performed as well as the operations which are permitted. All access control policies include constraints that limit their applicability. Constraints are in essence boolean expressions, expressed in terms of the local context, trust values or other information conveyed with the request in the form of tokens.

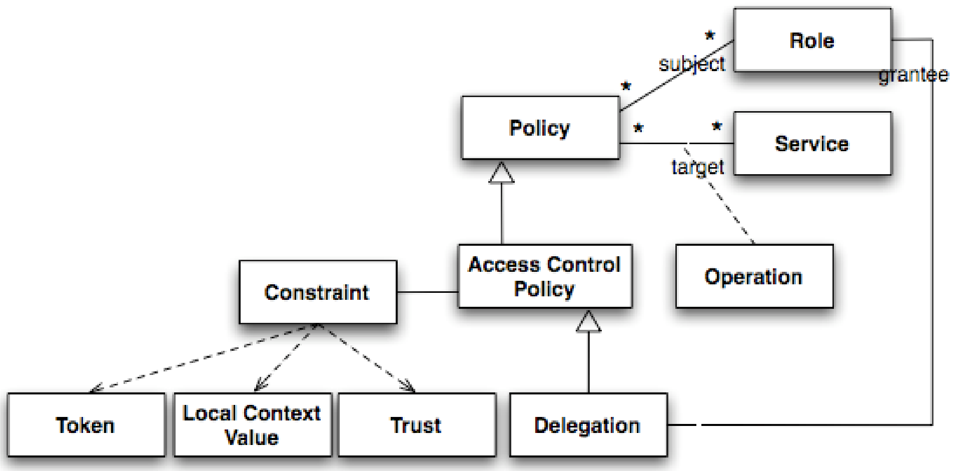

Figure 19 - Access Control Policies structure

Delegation policies express the granting of rights for the purpose of administering access control. This allows to decentralise administration closer to the decision maker and to differentiate between access permissions and administrative permissions. Thus delegation policies need to identify the *grantee* to which rights have been issued in order to specify additional access rights.

o Conclusions

The models described above present a simple yet powerful conceptual framework for specifying the access control and adaptation needs within the TrustCoM framework. In particular the obligation policies are generic and may be used in order to trigger:

- The execution of administrative business processes when participants enter or leave the VO.
- Reconfiguration actions on existing services including reconfiguration of the security parameters as a function of changing trust or reputations levels.
- Accounting or monitoring procedures in response to SLA violations or new user requirements.
- Instantiation, enforcement and removal of policies within the VO.

Relationships provide not only a grouping construct for policies but also a unit of deployment and distribution as described in D9 Architecture and D18 Framework specification.

7 Enterprise Network and Virtual Organisation Infrastructure (HLRS)

The provisioning and enactment of virtual organisations across enterprise boundaries as envisaged by TrustCoM requires a stable infrastructure capable of providing

- uniform messaging and notification support

  communication between participants should be straight-forward and observe the security, privacy and policy issues as described in the subsystem sections above.

- means of identifying potential participants

  in order to participate in a virtual organisation, service providers need to publish information about the service they offer, like functionalities and quality of service they can maintain

- statefulness

  as opposed to grid services, web services are in themselves not stateful, i.e. can not maintain state information between requests

- support for information contribution

  configuration of the services to the TrustCoM requirements, like subscription-information, endpoints reference addresses, policies, SLAs etc. (cf. above chapters) needs to be provided to the individual participants in a coordinated manner

To realise these issues implies certain restrictions on the design modelling and adds further requirements that shall be discussed in this chapter.

7.1 Discovery Support

Identification of partners according to their capabilities (and limitations) is a requirement that has been identified early in the project to allow for a means to discover services that may fulfil the respective business tasks. Such a concept has already been developed by the UDDI specifications [REFERENCE] and TrustCoM will not significantly change this *concept*, however extensions may be made to the specification according to the information needs identified in TrustCoM that go beyond the UDDI capabilities (cf. D9, D18)

Note that provisioning of additional information about a service provider (quality of service, trustworthiness etc.) are covered in the subsystem sections above.

7.2 Virtual Service Nodes

In order to realise a uniform messaging (and notification) infrastructure, TrustCoM pursues the concept of "virtualised" service nodes:

These Virtual Service Nodes act in principal as interfaces to the (application) services, thus serving as a interceptor capable of enacting the TrustCoM relevant issues upon in- and outgoing messages (see below) and providing notification support. Ideally, such a "proxy" allows for simple "plug & play", i.e. any service provider may become "TrustCoM-enabled" simply by installing or subscribing to a proxy of this sort. Whilst this is obviously restricted by implementation issues, the scenario is nonetheless valid as a concept to drive programming accordingly.

A "virtualised node" will on the one hand enhance the operations provided to a customer, on the other hand provide features to the service provider that it would otherwise not support – from the TrustCoM point of view, these enhancements cover in particular:

- Manageability

  that will allow accessing status information about the service, respectively resource and optionally provides the capability to manage the service by exposing specific methods, like e.g. resetting the status

- Security

  authenticating in- (and out-going) messages, thus ensuring that VO members can be unequivocally identified (cf. section 5)

- Policy

  verifying the access rights of the sender and potentially requesting to block the message (cf. section 6)

- SLA

  hosting the monitoring service and potentially a pre-evaluator to assess the system state with respect to the quality of service (cf. section 3)

These concepts, with exception from Manageability which will be discussed below, have already been covered in preceding sections.

From the customer's point of view, the service *itself* actually provides these features, as he/she can not distinguish between the virtualised node and the actual node itself.

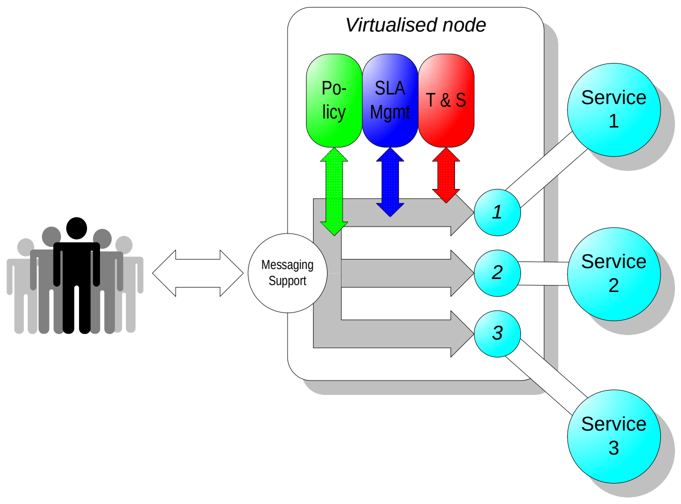

Figure 20 - the virtualised service node concept

7.2.1 Messaging and notification support

In summary, the virtual node acts as a gateway or interceptor to messages between customer and service, thereby enacting the above listed features. It is therefore capable of filtering messages according to policy- and/or security issues, e.g. if the sender lacks the required access rights (cf. the respective sections for more details)

The "proxy" may also provide some parsing functionality from the messaging standard enacted between virtual node and customer to the standard used by the service. Thus it is theoretically possible to support a wide range of standards used by different services, while still enacting a uniform messaging standard between different virtual nodes, as would be the case in a virtual organisation.

Similarly, the "proxy" may allow automatic handling of notification subscriptions and productions, in particular where related to the enhanced functionalities provided by the virtual node.

Note that the virtual node may maintain a log of important messages for later reference.

7.2.2 Localisation

Since the virtual node "hides" the actual service from the customer, it needs to be capable of forwarding messages to the right endpoint, i.e. of acting as a broker. Accordingly, it will maintain a list of actual "locations" of the services it serves, which may be updated with changes, e.g. when the actual service moves to a different address.

Thus, the virtual nodes acts as some kind of "handle resolver" that may map the destination reference of the customer to the actual endpoint of the respective service.

7.2.3 Manageability

As a representative of the actual service, the virtualised node may expose information about it that may be used for management and discovery purposes.

Likewise the proxy may provide information about the service like the quality of service it may principally maintain, what actions it can perform etc., i.e. data that will be queried when a service for a specific purpose and with specific parameters is searched.

What is more, the proxy may expose this information during run-time, i.e. to represent *current* information about the service with respect to its performance and its status – this allows, amongst others, monitoring the quality of service. In case of the HPC provider of TrustCoM's collaborative engineering scenario, such status and performance information may cover whether a cluster is currently available or occupied, a list of jobs queued for the machine, the current CPU loads, working temperature etc.

By allowing not only to view the respective data, but also to manipulate it, e.g. adding or removing jobs from the queue, the conveying virtual node becomes a "manageability interface" to the service, i.e. an enhancement of the service's normal functionality in such a way as to get and set status-related data.

7.3 Statefulness

As opposed to the "plain" web service concept, the Grid concept foresees "stateful" services, i.e. processes that can retain information of some form between two requests, which is strongly desired not only by TrustCoM, but by most enterprises exploiting web service technologies. This concept has been successfully introduced to the web service domain quite a while ago by such specifications as WSRF and WS-Transfer and shall not be examined here further.

Consequently, for the TrustCoM project it is of main interest how these concepts will be *used*:

Similar to the Grid Service concepts we foresee flexible VO setups by allowing for dynamic instantiation of "stateful" resources (services) at any given location. Thus TrustCoM supports the concept of a service "factory", which will allow dynamic instantiation of services at various locations. With respect to web services, this really involves the following issues:

- moving the code to the designated location
- setting up a virtual directory, thus allowing access to the code
- instantiating a web service resource for statefulness and
- linking it to the web service

7.4 Implementation issues

It should be noted that a virtual node as depicted above encounters significant implementation and practical restrictions, as (1) the actual status information still needs to be made available by the service provider itself and (2) no service provider would really want to leave almost complete control over the service to a third party entity. Thus, within the TrustCoM project, we shall take an approach that tries to maximise the functional benefits from above described framework, whilst minimising the involved security risks and leaving complete control up to the service provider.

Amongst other issues, this involves keeping the “virtual node” within the service provider’s domain and splitting up the messaging and notification support into distributed entities that are supported by (trusted) third party “brokers” in order to allow for subscription management and message forwarding. The details with respect to (code-able) components, however, shall be discussed in the architecture deliverable D9.

Similar issues apply to dynamic instantiation: a Service Provider generally will not want to let service instantiation and choice of hosts out of his/her hands, thus the actual instantiation may be influenced by the VO only to a limited degree. However, given this concept, implementation efforts are under way to put the *capabilities* of a service factory at the service provider’s disposal that may be used to his or her discretion.

7.5 Conclusion

The Infrastructure concepts express some high-level “guidance” towards implementation that is however restricted by implementation issues. The main ideas pursued from an infrastructure point of view are:

- virtualised service nodes that act on behalf of the service and hide it from direct access by the customer
- uniform messaging and notification support between participants of the VO
- dynamic VO structures by allowing flexible service instantiation
- autonomous localisation tracking of involved participants
- exposure of service information and provision of manageability methods
- stateful web services between requests

8 Conclusion and Future Plans

This document is one of four contributing to the Trustcom Framework. The others address the overall Framework, the architecture to implement it, and generic methods and tools to support it. The objective of this document was take outline the conceptual models underlying the Framework.

What this document has not done is to describe the details of the languages to be used to represent Contracts, Service Level Agreements, Policies or Recommendations. It has also not described the processes of refinement that derive one from another, or the algorithms to reason over the monitored policies at run time. All of these will be defined in the next update of the conceptual models in six months time.