Large Scale Integrating Project

Grant Agreement no.: 257899

# D8.1 – Social ontologies, protocols and APIs report and specification

**SMART VORTEX –WP8-D8.1**

| | |
|---|---|
| Project Number | FP7-ICT-257899 |
| Due Date | 30.09.2012 |
| Actual Date | 26.09.2012 |
| Document Author/s: | Jens Grabarske, Felix Engel, Dominic Heutelbeck |
| Version: | 1.0 |
| Dissemination level | RE |
| Status | Draft M24 |
| Contributing Sub-project and Workpackage | WP8 |
| Document approved by | John Lindström |

Co-funded by the European Union

**Document Version Control**

| Version | Date | Change Made (and if appropriate reason for change) | Initials of Commentator(s) or Author(s) |
|---|---|---|---|
| 0.1 | 04.07.2012 | Initial structural Draft | JG, FE |
| 0.2 | 14.09.2012 | Added ontology descriptions | JG, FE |
| 0.3 | 20.09.2012 | Added roadmap and summary, applied some recommendations | JG, FE |
| 0.4 | 26.09.2012 | Added appendices | JG |
| 1.0 | 28.09.2012 | Last changes | JG, DH |

**Document Change Commentator or Author**

| Author Initials | Name of Author | Institution |
|---|---|---|
| DH | Dominic Heutelbeck | FTK e.V. |
| JG | Jens Grabarske | FTK e.V. |
| FE | Felix Engel | University of Hagen |
| JL | John Lindström | Luleå Tekniska Universitet |

**Document Quality Control**

| Version QA | Date | Comments (and if appropriate reason for change) | Initials of QA Person |
|---|---|---|---|
| 1.0 | 28.09.2012 | | JL |
| 1.0 | 30.09.2012 | Minor changes | IK |
| | | | |

**Catalogue Entry**

| | |
|---|---|
| **Title** | Social ontologies, protocols and APIs report and specification |
| **Creators** | Jens Grabarske, Felix Engel, Dominic Heutelbeck |
| **Subject** | |
| **Description** | Description of the progress on WP8. |
| **Publisher** | |
| **Contributor** | |
| **Date** | |
| **ISBN** | |
| **Type** | |
| **Format** | |
| **Language** | English |
| **Rights** | |

**Citation Guidelines**

## EXECUTIVE SUMMARY

This deliverable details the work on tasks 8.1-8.7, especially the development status of the social data models (tasks 8.1, 8.7) and the protocols and APIs used for integration of the components of the other partners (tasks 8.2-8.6, 8.8-8.9).

The models developed by FTK and FUH for tasks 8.1 and preliminary work on 8.7 are finished and already form the basis of the reference implementations. The approach used is known in literature as “institutional rules” and defines membership in organisational structures. The only open issue is the integration of the models which will be achieved in Year 2 and to form a complete model of all industrial partners in 8.7.

Tasks 8.2, the social communication protocol, are almost complete in the sense that the current implementation can already be used to access the social artefacts. For task 8.3, the group collaboration API, there is currently work between the partners to decide on the most convenient way of combining the efforts there with task 8.2 as prerequisite of task 8.4, the services for embedding collaboration tools. The protocol is transport agnostic, meaning that it can be used with an arbitrary transport protocol, but the transported packages need to conform to a format dictated by the embassy abstraction used in the project.

As the deadline of D11.2 was pushed back to M36, a full integration with the rules and policy framework for task 8.5 could not yet be accomplished.

## GLOSSARY

**CCO**
An ontology (see below) for corporate terms using the GSO, defined as part of WP8

**GSO**
An upper ontology for social constructs, task 8.1

**Ontology**
A formal, explicit specification of a shared conceptualisation [GRU93]

**REST**
A style of software architecture for distributed systems. Web service calls are sessionless HTTP requests that are idempotent. [FIE00]

**SAO**
Abbreviation for *service-oriented architecture*, a software architecture where components are seen as individual services used by other components. Common architecture for distributed systems.

**Semantic annotation**
Adding semantic information to data sets used by the system. By using ontologies machine-readable information can be added that refers to the meaning or overall semantics of a request, rather than just a syntactic exchange. [ORE06]

**Social ontology**
An ontology for social terms.

**Upper layer ontology**
Ontologies that define very general concepts to serve as common ground for the communication across domains [NIL01]

**XMLRPC**
A webservice protocol, more light-weight than SOAP, mostly using HTTP as transport.

# TABLE OF CONTENTS

# 1. INTRODUCTION AND OBJECTIVES

Social networking components are an essential part of the broader vision of SMART VORTEX. As such, they need to be carefully designed and considered to achieve the goal of cross-organisational collaboration and cooperation.
This deliverable details the work done on tasks 8.1 – 8.4 and 8.7. It defines the social communication protocols to be used and upper layer ontologies for social graphs, able to express cross-organisational business structures. Domain ontologies are further discussed as part of the skill modelling done by FUH.

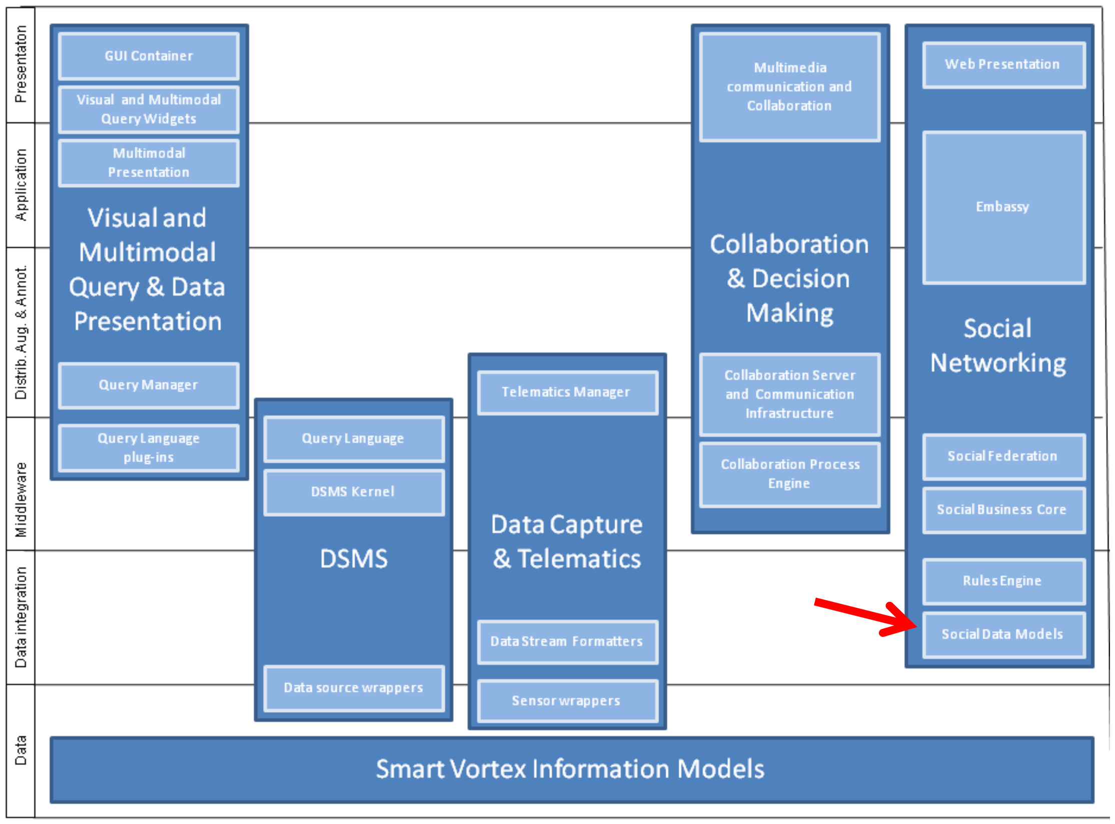

**Figure 1: The SMART VORTEX architecture as part of D2.3**

As such this deliverable details the work done on the Social Data Models. While this work does not have output that is directly measurable in terms of KPIs as defined in D1.2, it is a necessary requirement for the work on the semantic social collaboration suite as described in D8.2.

# 2. SOCIAL ONTOLOGIES

Collaboration sessions and collaborative processes are always team efforts and as such social activities that involve a group of people. In the application domain of Smart Vortex, these teams not only span over different project groups but also organisational boundaries. On the one hand, every participating organisation has a social structure that is distinct for this party. On the other hand, the collaborating group itself has a structure.
Therefore the need arises to communicate the different representations of social structures throughout the collaborative network. In Smart Vortex this is done through the use of social ontologies and social protocols. The former are the language with which the different instances can exchange social information, the latter are the protocols used for this exchange.

Ontologies usually have dependencies on other ontologies to reuse verbs and nouns defined by them. Ontologies that are meant to be included and not to be used directly are called *upper layer ontologies*. For Smart Vortex, several of these were defined to express all the necessary properties in social structures. The first and most important is the *general social ontology* (GSO) that is used to define terms like agent, social construct or membership. The second ontology that was defined was used to add special properties to the ontology and to further detail certain constructs. This was called the *social archetype ontology* (SAO). Finally, the *common corporate ontology* (CCO) allows for more detailed descriptions on the enterprise level. A graph showing the concepts and properties introduced by all three ontologies can be found in appendix A.
There are a couple of pre-existing and widely used upper layer ontologies.

## 2.1. General Social Ontology (GSO)

From the point of view of the GSO, every social context can be modelled in the same way. This assumption is based on the observation that there are two major models that can be used to model social context, “social relations” and “institutional rules” [KAO05] “Social relations” model the agents of a social structure as nodes of a graph and the interpersonal relationships as vertices. Larger structures (e.g. teams or departments) are not modelled explicitely but implicitely. For example: all the members of a team have a relationship with the team lead that puts him into a position of authority.
“Institutional rules” model team and department directly by assigning roles to participants in a social context. For example: in a team, someone is labelled “team lead”, someone is labelled “developer” and so on. This approach is very common in organisational ontologies. For example, the W3C working draft “An organization ontology” [W3C12] follows much of the same reasoning as the GSO and some structures are strikingly parallel even though they were developed independently [EPI12].
In a paper [GRA12] we could show that social relations and institutional rules can model the same constructs as the other, but institutional rules are easier to implement. Therefore the GSO utilises this approach to model social constructs.

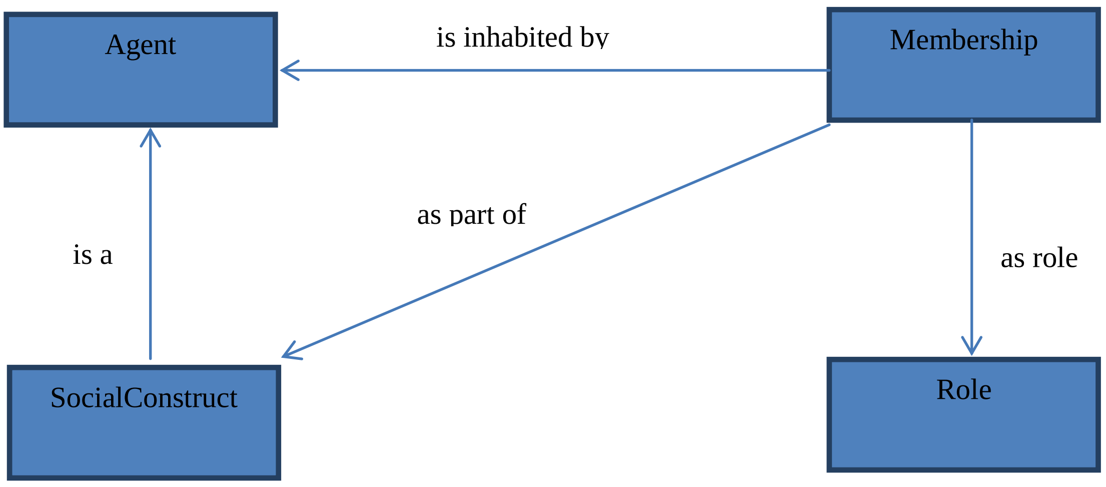

Figure 2: Representation of social constructs in the GSO

The illustration above gives a rough overview of how institutional rules are implemented.

An agent is an “acting entity” meaning it can be a person (more importantly: a user), a software program or any other entity that is able to cause activities within the system.
A social construct is a special agent that consists of other agents. Agents can be associated to social construct by instances of the class *Membership* that also assigns a role to this participation.

For example: to express that Alice is team lead of a team called *Example Team*, a membership instance would be created that is associated to Alice via the “isInhabitedBy” property, to *Example Team* via the “asPartOf” property and to the role “team lead” via the “asRole” property.

The *Membership* class is a way to reify the relationship between the participating agent and the social construct to annotate it – especially with the role. This mechanism has its drawbacks, most importantly that it is hard to express further constraints of the social construct. For this reason, the *social archetype ontology* was developed.

The GSO can be found in appendix B in Turtle notation.

## 2.2. Social Archetype Ontology (SAO)

Social constructs differ in certain behavioural properties. To name a popular example, someone can become a follower on Twitter without the consent of the one who is followed, while contact requests on Facebook always require an affirmation. These behavioural properties are called *social properties* and they are defined as features of the social construct. Currently social properties are defined as tuples (c,lr,rr,t,lv,rv) where

| | |
|---|---|
| • c | is the social construct in question |
| • lr | is the *local role*, i.e., the role of the object checking for the behaviour |
| • rr | is the *remote role*, i.e., the role of the person to whom the behaviour is directed |
| • t | is the *type* of the social property |

- lv is the *local value* of the property (either 0 or 1 resp. false or true)
- rv is the *remote value* of the property (either 0 or 1 resp. false or true)

So far we have identified the following types of social properties:

- **Acceptance**
  If the local value is false and the remove value is true, someone with the local role in the social construct or someone who wants to attain the local role in the construct must ask permission of someone with the remote role in the construct. A local value of "true" means that the agent does not need to ask for permission. A remote value of "true" means that the other agent may reject the participation of the local agent in the construct.

- **Visibility**
  Determines who can see whom. A local value of "false" and a remote value of "true" means that the local agent cannot see the remote agent, but the remote agent can see the local agent.

- **Distribution**
  Determines who receives resources from whom. A local value of "false" and a remote value of "true" means that messages are sent from the remote agent to the local agent, but not vice versa.

Apart from these properties, the SAO also contains properties to determine the arity of a construct.

The SAO can be found in appendix C in Turtle notation.

## 2.3. Common Corporate Ontology (CCO)

Using the SAO and the GSO, the CCO introduces some important concepts for corporate structures, like *organization, group* or *crowd*.
Here, *organization* is any social construct that is ordered, i.e., has a hierarchical structure. A *crowd* is the opposite, a construct where no clear responsibilities exist. Furthermore, properties for hierarchical concepts are introduced.

The CCO can be found in appendix D in Turtle notation.

# 3. SOCIAL PROTOCOLS

The reference implementation of the SMART VORTEX suite can be accessed using REST and XMLRPC using the embassy mechanism developed by Jens Grabarske [GRA11]. All calls to the business end of the social suite (called SBC for Social Business Core) are asynchronous. To allow for callback mechanisms, callers must have a server of their own to receive a reply. Alternatives like Comet are currently being considered but aren't implemented yet.

## 3.1. API specification

For cross-organizational and implementation-agnostic communication it is essential to have a common API with which agents can be registered, constructed or deleted. This way the federated systems can synchronise the state of their social networks.
The current state of the API is still a work in progress but already includes all necessary functionality for cross-organizational data exchange. Whether or not actions need to be altered or added is subject to the practical tests and surveys to be conducted in Year 3.
The following actions are provided with their respective arguments and can be called using the mechanisms of the used connector:

- http://www.openconjurer.org/ontologies/gso.owl#joinSocialConstruct
  Arguments:

| | | |
|---|---|---|
| 0 | agent | the agent to join the construct |
| 1 | construct | the construct to join |
| 2 | role | the role to be assigned to the agent |

  This action, if successful, makes the agent a member of the social construct inhabiting the given role, provided, the properties and policies allow it.

- http://www.openconjurer.org/ontologies/gso.owl#leaveSocialConstruct
  Arguments:

| | | |
|---|---|---|
| 0 | agent | the agent to leave the construct |
| 1 | construct | the construct to be left |

  If successful, the agent is no longer a member of the construct.

- http://www.openconjurer.org/ontologies/gso.owl#setMembershipRole
  Arguments:

| | | |
|---|---|---|
| 0 | agent | an agent |
| 1 | construct | the construct in question |
| 2 | role | a role |

  If successful, the role of the agent within the social construct is changed.

- http://www.openconjurer.org/ontologies/gso.owl#removeAgent
  Arguments:

| | | |
|---|---|---|
| 0 | agent | an agent |

If successful, the agent is removed from the social network entirely.

- http://www.openconjurer.org/ontologies/gso.owl#createAgent
  Arguments:

| | | |
|---|---|---|
| 0 | string | ontology class URI of the type |
| 1 | string | Preset ID (optional) |
| 2 | agent | home organization (optional) |

This creates a new agent instance in the social network. If the agent is added to the social network to represent a remote or foreign agent, the ID in the other network can be given as well as the URI of the owning organisation. Otherwise the home organisation of the local network will be assumed and a random ID will be generated.

- http://www.openconjurer.org/ontologies/gso.owl#inviteToConstruct
  Arguments:

| | | |
|---|---|---|
| 0 | construct | the social construct |
| 1 | agent | the agent to be invited |
| 2 | role | the role to be assigned |
| 3 | boolean | whether the invite is founding (optional) |

This invites an agent to a social construct. If successful, the agent will have a membership with the given role. If the social properties demand that the agent needs to affirm the membership, this is enforced by sending the agent an invite, which is a social event. If the social construct is marked as founding (default is false), then the social construct will be deleted if the invite is refused.

- http://www.openconjurer.org/ontologies/gso.owl#getAllEvents
  No arguments.

This returns all the social events for the initiating agents, including invites.

- http://www.openconjurer.org/ontologies/gso.owl#acceptInvite
  Arguments:

| | | |
|---|---|---|
| 0 | invite | the invite to be accepted |

This action informs the social business core that the invite was accepted and that the pending join of the social construct is to be performed.

- http://www.openconjurer.org/ontologies/gso.owl#denyInvite
  Arguments:

  |  |  |  |
  |---|---|---|
  | 0 | invite | the invite to be denied |

  This action informs the social business core that the invite was denied. The inviters are informed using a social event. If the social construct was founding, then it will be deleted.

All actions are subject to policies and the behavioural control of the social properties.

# 4. SKILL MODELS IN ORGANIZATIONAL DOMAIN ONTOLOGIES

The SMART VORTEX domain model focuses on concepts that are required for cross organizational communication, with focus on the representation of social relations. Another aspect of this model is the introduction of concepts that supports the representation of relations between professionals. Such relations are crucial e.g. for the initiation of a collaborative work within problem solving tasks required within the SMART VORTEX industrial application scenarios (c.f. SMART VORTEX D9.1 section *Expert search in organizational models*). Central for the representation of such relations is the introduction of the skill concept and its embedding within an organization context.
Several ontologies exist already that include a very general organizational model or at least introduce the concept of an organization. FOAF[1] for instance claims to link people and information and provide a *foaf:Organization* class to link people to the organization they are belonging to. However, such ontologies are too generic and do not provide concepts required to meet the above requirements. Hence, existing organizational models will be reviewed with respect of its ability to model skills within organizations. Following that, a review of skill frameworks and taxonomies will be undertaken. Review findings finally will serve as input of knowledge and skill related concepts to the SMART VORTEX domain model.

## 4.1. Organizational Models

Several ontologies exist that explicitly model comprehensive various organizational aspects, and shall by reviewed as stated above.
An organization in the sense of the Toronto Virtual Enterprise project (TOVE) is defined according to Weber "*We consider an organization to be a set of constraints on the activities performed by agents."*[FBG96]. n organization hence consists of a set of Agents that are organized in Units with the intention to accomplish the Organization-Goals.  Agents could be either human or software entities, linked via Roles to the organization goals to be accomplished. Such Roles in fact are a set of functional requirements and constraints, that are needed to accomplish a specific goal.  Furthermore a Role is associated with Processes, Resources, Authority, Skills and Policies. Processes are activity networks, defined within the TOVE Activity Ontology and Resources within the TOVE Resource ontology. While the concept of *Goal*'s and *Authority* are part of the organizational model, *Skill* and *Policy* are not explicitly modeled in the organizational ontology. The *Position* concept is used as link between *Roles* and *Persons*, reflecting a formal employment position within the organization.

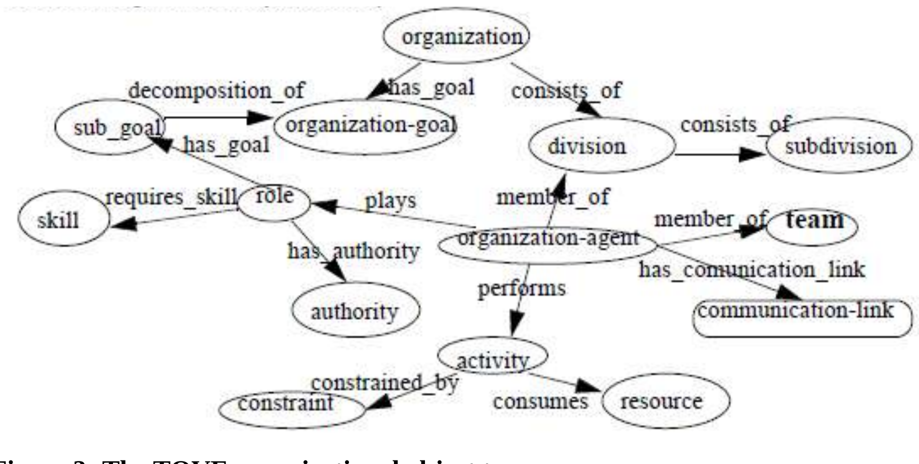

**Figure 3: The TOVE organizational object taxonomy**

[1] http://xmlns.com/foaf/spec/

A rather similar ontology has been developed as part of the Enterprise Project, the Enterprise Ontology organizational model defines a set of crucial concepts within an Organization. Important part within this organization model takes the notion of an Organizational Unit that defines legal entities and machines on the one side as well as structural elements of an organization on the other side.  Skill is part of the capability concept that describes a relationship between an Actor and an Activity. In this sense a Skill is potentially taken by a Person and its ability must be practiced or demonstrated to some measurable degree [**¡Error! No se encuentra el origen de la referencia.**].

The Open Group Architecture Framework (TOGAF) differs from the above approaches and is described as *is a detailed method and set of supporting tools – for developing enterprise architecture*[2]. In its core intended to support enterprise architects, by the provision of a process description (ADM Architecture Development Method) that manages the lifecycles of enterprise architectures. TOGAF considers an Organizational Unit as: *A self-contained unit of resources with goals, objectives, and measures. Organization units may include external parties and business partner organizations.* [http://pubs.opengroup.org/architecture/togaf9-doc/arch/chap34.html]. Actors are part of such an organization that interacts with some activities, while having a specific Role. A Role in turn is defined as the *expected function of an author* and the *contribution they make through the application of their skills, knowledge, experience and abilities*[3]. In TOGAF skills specifies roles of enterprise architects, though every role has several skills whereat each skill has one proficiency level. TOGAF specifies four levels of proficiency, ranging from background to expert.
Further relevant work has been also done during the PROTON Ontology specification developed during the SEKT[4] project as well as during the integrated project SUPER[5] or within the AKT[6] project.

Furthermore relevant to mention the Zachman Framework which is comprehensive enterprise architecture ontology that specifies essential components of an enterprise. In fact the Zachman framework is represented via a two dimensional matrix (*fundamentals of communication* and *Reification*) applicable for enterprise component specification. Since Zachman is a fairly comprehensive representation of enterprise concepts, it is seen as one of the most complete representation of enterprise concepts [**¡Error! No se encuentra el origen de la referencia.**]. However, the Zachman Framework is not a methodology that gives guidelines for the realization of enterprise object, but furthermore the ontology describing the enterprise components. Hence, the Zachman framework has to be mentioned for completeness, but could rather be used as starting point for a modelling work than instantiated as above mentioned ontologies. A rather similar approach has also been made by CIMOSA[7]. CIMOSA has been developed by the AMICE consortium intended as architecture for enterprise integration. Though it emphasis on a reference architecture for enterprise modeling, modeling language and a common terminology.

## 4.2. Skill Models

The introduction of the skill concept is essential needed within the SMART VORTEX organizational model as introduced above. Review of existing organizational models shows that the definition of skills is required to model requirements to accomplish organizational goals and to reflect existing knowledge within the organization. Several Skill taxonomies and frameworks exist already and will be reviewed in the following subsections. However, several important concepts are tightly coupled with the notion of *Skill* and shall be defined to clarify terminology.

[2] http://pubs.opengroup.org/architecture/togaf9-doc/arch/
[3] http://pubs.opengroup.org/architecture/togaf9-doc/arch/chap03.html#tag_03_61
[4] http://www.sekt-project.com/
[5] http://www.ip-super.org/
[6] http://www.aktors.org/akt/
[7] http://www.cimosa.de/

In a first approach the notion of Winterton et al [**¡Error! No se encuentra el origen de la referencia.**] is used to give a notion of Knowledge, Skill and Competence:

> **Knowledge:** Knowledge is sometimes viewed as if it was a concrete manifestation of abstract intelligence, but it is actually the result of an interaction between intelligence (capacity to learn) and situation (opportunity to learn), so is more socially-constructed than intelligence.
>
> **Skill:** Usually the term skill is used to refer to a level of performance, in the sense of accuracy and speed in performing particular tasks ("skilled performance").
>
> **Competence:** Key competences are context-independent, applicable and effective across different institutional settings, occupations and tasks. These typically include basal competences, such as literacy, numeracy, general education; methodological competences, like problem solving, IT skills; communication skills, including writing and presentation skills; and judgment competences, such as critical thinking.

In fact *Knowledge, Skill* and *Competence* are tightly coupled and could not be regarded in isolation, butsince the focus in this section is on modeling Skills the above notion shall be sufficient to define terminology. In the following some skill taxonomies and frameworks will be introduced.

## 5. SUMMARY AND ROADMAP

In Year 3, the efforts of FUH and FTK will be further integrated and the protocols will be extended to include the collaboration tools and services as part of tasks 8.3 and 8.4. As the rules and policy framework described in D11.2 will be available M36, work on task 8.5 will be a top priority to allow for a more complete review. The KPIs discussed in D1.2 for WP8 all apply to the reference implementation discussed in D8.2. Apart from the TSN, the ES and the EGF KPIs, no tests could be performed yet as the first version was completed recently. In year 3 FUH and FTK plan to perform test surveys to determine their values.
The following KPIs of WP8 were specified in D1.2:

| KPI | Name | Measurable | Expected |
|---|---|---|---|
| MTIS | Mean Time to Incident Solution | Not yet | Y3 |
| MTAC | Mean Time to Analyse Causes | Not yet | Y3 |
| MTGC | Mean Time to Group Composition | Not yet | Y3 |
| CMTAC | Collaboration Mean Time to Analyse Causes | Not yet | Y3 |
| CMTD | Collaboration Mean Time to Decision | Not yet | Y3 |
| CPS | Collaboration Participant Satisfaction | Not yet | Y3 |
| MITA | Mean Time to Access | Not yet | Y3 |
| CPC | Collaboration Participant Commitment | Not yet | Y3 |
| TSN | Technology Support Number | Yes<br>93 *) | Y2 |
| CGS | Collaboration Group Size | Not yet | Y3 |
| GIN | Generated Idea Number | Not yet | Y3 |
| FIQ | Feasible Idea Quote | Not yet | Y3 |
| OCN | Outcome Clarify Number | Not yet | Y3 |
| ES | Expert Search | Yes | Y2 |
| EGF | Expert Group Formation | Yes | Y2 |
| PCN | Pattern of Collaboration Number | Not yet | Y3 |

The TSN, the technology support number is the number of operating systems supporting the semantic collaboration suite. Oracle certifies that Java and its enterprise systems of version 1.5.0 (which was used during the implementation) is supported by 93 operating systems [ORAJ] as determined through virtualised system configurations. Whether the suite eventually does not work on one of these operating systems due to some restrictions could not be tested yet.

The EGF could be measured as social constructs can be constructed. For the ES KPI see the discussion below.

For the modeling of skills, the following KPIs are applicable for the Year 3 review:

| Milestone | Description | KPI |
|---|---|---|
| MS 16 | Review of existing skill-models and frameworks | Expert Search (ES) |
| MS 16 | Specification of Skill related concepts for integration to SMART VORTEX domain ontologie for usage in problem solving tasks | Expert Search (ES) |
| MS 16 | Integration of Skill related concepts to SMART VORTEX domain ontology | Expert Search (ES) |

|  |  |  |
|---|---|---|
| MS 24 | Preparation of skill related test data from industrial partners for ontology instantiation | Expert Search (ES) |
| MS 24 | Instantiation of SMART VORTEX domain ontology with Skill related data | Expert Search (ES) |

Planning of Year 4 will depend on the survey that needs to be conducted as consequence of the survey for the reference implementation as discussed in D8.2.

As collaboration tools developed in WP8 capture socially contextualized provenance information during design activities, task 8.6 will develop schemes and APIs that allow annotating product data streams with these provenance information. These semantic annotations support further data exploration and visualization in decision making processes. In addition to the semantics of social collaboration data, also authorization specifications which are developed in WP11 will be annoated to product data streams allowing more fine-grained access right definitions.

## References

[GRA11] Jens Grabarske (2011). *Use cases and security requirements for data exchange between corporate social networks*

[GRA12] Jens Grabarske, Dominic Heutelbeck (2012). *An upper ontology for the social web*. In Proceedings of ASONAM 2012.

[FBG96] Fox, M. S., Barbuceanu, M., & Gruninger, M. (1996). *An organisation ontology for enterprise modeling : Preliminary cloncepts for linking structure and behaviour*, 29, 123-134.

[FIE00] R. T. Fielding, "Architectural Styles and the Design of Network-based Software Architectures", PhD thesis, University of California, Irvine, 2000

[LH01] Lagoze, C., Hunter, J. (2001) *The ABC Ontology and Model Purpose and Scope of the ABC Model*.

[KAO05] E. Kao, P. Chang, Y. Chien et al. *Using ontology to establish social context and support social reasoning*. Lecture Notes in Computer Science 2005. 344-357

[NIL01] I. Niles "Towards a standard upper ontology", International Conference on Formal Ontology, 2001

[ORE06] E. Oren, K. Hinnerk, S. Scerri, S. Handschuh, M. Sintek, "What are Semantic Annotations?", Artificial Intelligence, 2006

[WDS05] Winterton, J., Delamare - LeDeist, F., & Stringfellow, E. (2005). *Typology of knowledge, skills and competences: clarification of the concept and prototype*, (January).

[RA12] Rajabi, Z., & Abade, M. N. (2012). Data-Centric Enterprise Architecture. *International Journal of Information Engineering and Electronic Business*, *4*(4), 53–60. doi:10.5815/ijieeb.2012.04.08

[UKMZ98] Uschold, M., King, M., Moralee, S., & Zorgios, Y. (1998). The Enterprise Ontology, *13*(Ker 13108).

[EPI12] Epimorphics, "Organization Ontology – A series of developer postings on developing an organization ontology"

http://www.epimorphics.com/web/taxonomy/term/4/0

[GRU93] T. R. Gruber, "A translation approach to portable ontology specifications", Knowledge Acquisition 5, 1993, p. 199-220

[W3C12] W3C, "An organization ontology", W3C Working Draft 05 April 2012

http://www.w3.org/TR/vocab-org/

[Z12] The Zachman Framework, http://zachmaninternational.com/index.php/the-zachman-framework, visited 2012.09.19

[ORAJ] Java 2 Platform 5.0 and Java for Business 5.0

http://www.oracle.com/technetwork/java/javase/system-configurations-139801.html

## APPENDIX A: ONTOLOGIES MAP

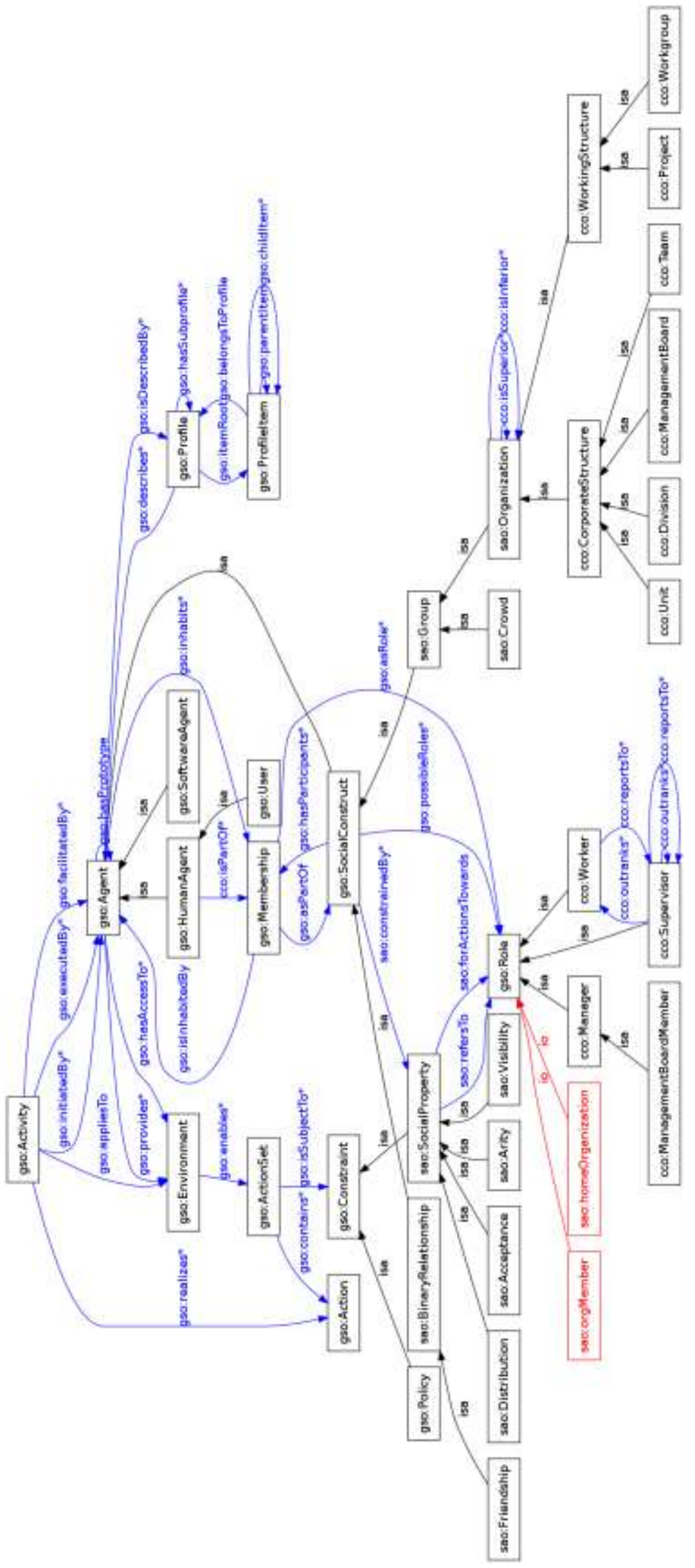

Figure 4: Complete graphical representation of the GSO, SAO and CCO

## APPENDIX B: GSO IN TURTLE NOTATION

```
@prefix rdfs:    <http://www.w3.org/2000/01/rdf-schema#> .
@prefix swrl:    <http://www.w3.org/2003/11/swrl#> .
@prefix protege:  <http://protege.stanford.edu/plugins/owl/protege#> .
@prefix default:  <http://www.openconjurer.org/ontologies/gso.owl#> .
@prefix xsp:     <http://www.owl-ontologies.com/2005/08/07/xsp.owl#> .
@prefix swrlb:   <http://www.w3.org/2003/11/swrlb#> .
@prefix xsd:     <http://www.w3.org/2001/XMLSchema#> .
@prefix owl:     <http://www.w3.org/2002/07/owl#> .
@prefix rdf:     <http://www.w3.org/1999/02/22-rdf-syntax-ns#> .

default:involves
    rdf:type owl:ObjectProperty .

default:ProfileItem
    rdf:type owl:Class .

default:Agent
    rdf:type owl:Class ;
    rdfs:comment "<p style=\"margin-top: 0\">\n         Most often, when people use the term
`agent' they refer to an entity \n          that functions continuously and autonomously in an
environment in which \n       other processes take place and other agents exist. (Shoham1993)\n
</p>"@en .

default:contains
    rdf:type owl:ObjectProperty ;
    rdfs:domain default:ActionSet ;
    rdfs:range default:Action .

default:describes
    rdf:type owl:ObjectProperty ;
    rdfs:domain default:Profile ;
    rdfs:range default:Agent ;
    owl:inverseOf default:isDescribedBy .

default:inhabits
    rdf:type owl:InverseFunctionalProperty , owl:ObjectProperty ;
    rdfs:domain default:Agent ;
    rdfs:range default:Membership ;
    owl:inverseOf default:isInhabitedBy .

default:itemValue
    rdf:type owl:DatatypeProperty ;
    rdfs:domain default:ProfileItem ;
    rdfs:range xsd:string .

default:provides
    rdf:type owl:ObjectProperty ;
    rdfs:domain default:Agent ;
    rdfs:range default:Environment ;
    rdfs:subPropertyOf default:hasAccessTo .
```

```
default:parentItem
      rdf:type owl:FunctionalProperty , owl:ObjectProperty ;
      rdfs:domain default:ProfileItem ;
      rdfs:range default:ProfileItem ;
      owl:inverseOf default:childItem .

default:Action
      rdf:type owl:Class ;
      rdfs:comment "<p style=\"margin-top: 0\">\n      An action is an operation that either descibes the state change of a \n      machine by mapping a present state to another, or discloses data based \n      on the current state of a machine, or a mathematical function \n      independent of a resource store, or a combination of the other three. \n      The set of all actions is denoted by $\\mathcal{A}$ (Grabarske2011)\n    </p>"@en .

default:modificationDate
      rdf:type owl:DatatypeProperty .

default:Role
      rdf:type owl:Class ;
      rdfs:comment "<p style=\"margin-top: 0\">\n      A role is a capacity in which an agent acts in a certain context. For \n      the execution of an activity, &#8220;initiator&#8221;, &#8220;facilitator&#8221; and &#8220;actor&#8221; are \n      roles for participating agents [&#8230;] (Grabarske2011)\n    </p>"^^xsd:string .

default:creationDate
      rdf:type owl:DatatypeProperty .

default:HumanAgent
      rdf:type owl:Class ;
      rdfs:comment "<p style=\"margin-top: 0\">\n      An agent who is human.\n    </p>"@en ;
      rdfs:subClassOf default:Agent .

default:Activity
      rdf:type owl:Class ;
      rdfs:comment "<p style=\"margin-top: 0\">\n      An activity is the execution of an action. (Grabarske2011)\n    </p>"@en .

default:itemDescription
      rdf:type owl:DatatypeProperty ;
      rdfs:domain default:ProfileItem ;
      rdfs:range xsd:string .

default:hasAccessTo
      rdf:type owl:ObjectProperty ;
      rdfs:domain default:Agent ;
      rdfs:range default:Environment .

default:childItem
      rdf:type owl:InverseFunctionalProperty , owl:ObjectProperty ;
      rdfs:domain default:ProfileItem ;
      rdfs:range default:ProfileItem ;
      owl:inverseOf default:parentItem .
```

```
default:possibleRoles
    rdf:type owl:ObjectProperty ;
    rdfs:domain default:SocialConstruct ;
    rdfs:range default:Role .

default:itemType
    rdf:type owl:DatatypeProperty ;
    rdfs:domain default:ProfileItem ;
    rdfs:range xsd:string .

default:isSubjectTo
    rdf:type owl:ObjectProperty ;
    rdfs:domain default:ActionSet ;
    rdfs:range default:Constraint .

default:ActionSet
    rdf:type owl:Class ;
    rdfs:comment "<p style=\"margin-top: 0\">\n       A collection of actions. Used in scenarios
where it is easier or more \n      practical to talk about a group of actions rather than individual \n
actions.\n    </p>"@en .

default:SocialConstruct
    rdf:type owl:Class ;
    rdfs:comment "<p style=\"margin-top: 0\">\n       A social construct is a set of agents that
engage in social actions, \n       i.e., actions that either involve more than one agent or which
change or \n      interact with another agent or both.\n    </p>"^^xsd:string ;
    rdfs:subClassOf default:Agent .

default:Policy
    rdf:type owl:Class ;
    rdfs:comment "<p style=\"margin-top: 0\">\n       A policy is a definite course or method of
action selected from among \n       alternatives and in light of given conditions to guide and
determine \n      present and future decisions. (MerriamWebster)\n    </p>"@en ;
    rdfs:subClassOf default:Constraint .

default:User
    rdf:type owl:Class ;
    rdfs:comment "<p style=\"margin-top: 0\">\n       A user is a human being who triggers
activities within the middleware \n      and is therefore an agent. (Grabarske2011)\n    </p>\n\n<p
style=\"margin-top: 0\">\n       Note that not every human agent is necessarily a user. Other
humans may not be direct users of the\n      system but their activities need to be logged as
well.\n</p>"@en ;
    rdfs:subClassOf default:HumanAgent .

default:itemRoot
    rdf:type owl:FunctionalProperty , owl:ObjectProperty ;
    rdfs:domain default:Profile ;
    rdfs:range default:ProfileItem .

default:Environment
    rdf:type owl:Class ;
    rdfs:comment "<p style=\"margin-top: 0\">\n       Any larger body of entities that are related
to each other and in which \n       agents can exist or be situated in a meaningful way.\n
</p>"@en .
```

```
default:realizes
    rdf:type owl:ObjectProperty ;
    rdfs:domain default:Activity ;
    rdfs:range default:Action .

default:isInhabitedBy
    rdf:type owl:FunctionalProperty , owl:ObjectProperty ;
    rdfs:domain default:Membership ;
    rdfs:range default:Agent ;
    owl:inverseOf default:inhabits .

default:Membership
    rdf:type owl:Class ;
    rdfs:comment "<p style=\"margin-top: 0\">\n      An abstract construct to mark the
membership of an agent in a social \n      construct. It is used to annotate the relation with the
role the agent \n      has within the construct.\n    </p>"^^xsd:string .

<http://www.openconjurer.org/ontologies/gso.owl>
    rdf:type owl:Ontology .

default:houses
    rdf:type owl:InverseFunctionalProperty , owl:ObjectProperty ;
    owl:inverseOf default:appliesTo .

default:hasParticipants
    rdf:type owl:InverseFunctionalProperty , owl:ObjectProperty ;
    rdfs:domain default:SocialConstruct ;
    rdfs:range default:Membership ;
    owl:inverseOf default:asPartOf .

default:SoftwareAgent
    rdf:type owl:Class ;
    rdfs:comment "<p style=\"margin-top: 0\">\n      A computer program that acts as an agent.\n
</p>\n    <p style=\"margin-top: 0\">\n      \n    </p>\n    <p style=\"margin-top: 0\">\n
<b>This is a different context than in multi-agent systems!</b>\n    </p>"@en ;
    rdfs:subClassOf default:Agent .

default:belongsToProfile
    rdf:type owl:FunctionalProperty , owl:ObjectProperty ;
    rdfs:domain default:ProfileItem ;
    rdfs:range default:Profile .

default:enables
    rdf:type owl:ObjectProperty ;
    rdfs:domain default:Environment ;
    rdfs:range default:ActionSet .

default:appliesTo
    rdf:type owl:FunctionalProperty , owl:ObjectProperty ;
    rdfs:comment "<p style=\"margin-top: 0\">\n      An activity always occurs within a given
environment.\n    </p>"@en ;
    rdfs:domain default:Activity ;
    rdfs:range default:Environment ;
    owl:inverseOf default:houses .
```

```
default:Profile
      rdf:type owl:Class ;
      rdfs:comment "<p style=\"margin-top: 0\">\n      An entity that represents a user in a certain
organisation. On the one \n      hand it is a visible entity that can be scrutinised by another agent,
on \n           the other hand its URI can be used to represent the user where several \n
representations are available.\n    </p>"@en .

default:roleName
      rdf:type owl:DatatypeProperty ;
      rdfs:domain default:Role ;
      rdfs:range xsd:string .

default:executedBy
      rdf:type owl:ObjectProperty ;
      rdfs:domain default:Activity ;
      rdfs:range default:Agent ;
      rdfs:subPropertyOf default:involves .

default:isDescribedBy
      rdf:type owl:ObjectProperty ;
      rdfs:domain default:Agent ;
      rdfs:range default:Profile ;
      owl:inverseOf default:describes .

default:asPartOf
      rdf:type owl:FunctionalProperty , owl:ObjectProperty ;
      rdfs:domain default:Membership ;
      rdfs:range default:SocialConstruct ;
      owl:inverseOf default:hasParticipants .

default:facilitatedBy
      rdf:type owl:ObjectProperty ;
      rdfs:domain default:Activity ;
      rdfs:range default:Agent ;
      rdfs:subPropertyOf default:involves .

default:passwordHash
      rdf:type owl:DatatypeProperty ;
      rdfs:domain default:Agent .

default:isPrototype
      rdf:type owl:DatatypeProperty ;
      rdfs:comment "Signals that his agent acts only as prototype and nothing else, i.e., is not a
\"real\" agent. This is important for searches or user lists."@en ;
      rdfs:domain default:Agent ;
      rdfs:range xsd:boolean .

default:hasSubprofile
      rdf:type owl:TransitiveProperty , owl:ObjectProperty ;
      rdfs:domain default:Profile ;
      rdfs:range default:Profile .
```

```
default:initiatedBy
      rdf:type owl:ObjectProperty ;
      rdfs:domain default:Activity ;
      rdfs:range default:Agent ;
      rdfs:subPropertyOf default:involves .

default:profileName
      rdf:type owl:DatatypeProperty ;
      rdfs:domain default:Profile ;
      rdfs:range xsd:string .

default:name
      rdf:type owl:DatatypeProperty ;
      rdfs:range xsd:string .

default:userName
      rdf:type owl:DatatypeProperty ;
      rdfs:domain default:Agent .

default:hasPrototype
      rdf:type owl:FunctionalProperty , owl:TransitiveProperty , owl:ObjectProperty ;
      rdfs:comment "<p style=\"margin-top: 0\">\n      A prototype is an instance that acts as a
blueprint for the current \n      instance. This means that all properties not defined by the current
\n      instance are derived from the prototype.\n    </p>\n    <p style=\"margin-top: 0\">\n      \n
</p>\n    <p style=\"margin-top: 0\">\n      An instance can only have one prototype, a prototype
can itself have \n      further prototypes.\n    </p>"@en ;
      rdfs:domain default:Agent ;
      rdfs:range default:Agent .

default:Constraint
      rdf:type owl:Class ;
      rdfs:comment "<p style=\"margin-top: 0\">\n      Any rule that constrains the possible actions
or their parameters.\n    </p>"@en .

default:asRole
      rdf:type owl:ObjectProperty ;
      rdfs:domain default:Membership ;
      rdfs:range default:Role .
```

## APPENDIX C: SAO IN TURTLE NOTATION

```
@prefix rdfs:    <http://www.w3.org/2000/01/rdf-schema#> .
@prefix swrl:    <http://www.w3.org/2003/11/swrl#> .
@prefix protege:  <http://protege.stanford.edu/plugins/owl/protege#> .
@prefix default:  <http://www.openconjurer.org/ontologies/sao.owl#> .
@prefix xsp:     <http://www.owl-ontologies.com/2005/08/07/xsp.owl#> .
@prefix swrlb:   <http://www.w3.org/2003/11/swrlb#> .
@prefix xsd:     <http://www.w3.org/2001/XMLSchema#> .
@prefix owl:     <http://www.w3.org/2002/07/owl#> .
@prefix gso:     <http://www.openconjurer.org/ontologies/gso.owl#> .
@prefix rdf:     <http://www.w3.org/1999/02/22-rdf-syntax-ns#> .

default:homeOrganization
    rdf:type gso:Role .

default:Friendship
    rdf:type owl:Class ;
    rdfs:subClassOf default:BinaryRelationship .

<http://www.openconjurer.org/ontologies/sao.owl>
    rdf:type owl:Ontology ;
    owl:imports <http://www.openconjurer.org/ontologies/gso.owl> .

default:Distribution
    rdf:type owl:Class ;
    rdfs:subClassOf default:SocialProperty .

default:Visibility
    rdf:type owl:Class ;
    rdfs:subClassOf default:SocialProperty .

default:remoteValue
    rdf:type owl:DatatypeProperty ;
    rdfs:domain default:SocialProperty ;
    rdfs:range xsd:boolean .

default:minArity
    rdf:type owl:FunctionalProperty , owl:DatatypeProperty ;
    rdfs:domain gso:SocialConstruct ;
    rdfs:range xsd:int .

default:forActionsTowards
    rdf:type owl:FunctionalProperty , owl:ObjectProperty ;
    rdfs:domain default:SocialProperty ;
    rdfs:range gso:Role .

default:Acceptance
    rdf:type owl:Class ;
    rdfs:subClassOf default:SocialProperty .
```

```
default:localValue
      rdf:type owl:DatatypeProperty ;
      rdfs:domain default:SocialProperty ;
      rdfs:range xsd:boolean .

default:constrainedBy
      rdf:type owl:ObjectProperty ;
      rdfs:domain gso:SocialConstruct ;
      rdfs:range default:SocialProperty .

default:orgMember
      rdf:type gso:Role .

default:Arity
      rdf:type owl:Class ;
      rdfs:subClassOf default:SocialProperty .

default:SocialProperty
      rdf:type owl:Class ;
      rdfs:subClassOf gso:Constraint .

default:BinaryRelationship
      rdf:type owl:Class ;
      rdfs:subClassOf gso:SocialConstruct .

[]    rdf:type owl:DataRange ;
      owl:oneOf ("neither"^^xsd:string "local"^^xsd:string "remote"^^xsd:string
"both"^^xsd:string) .

default:Group
      rdf:type owl:Class ;
      rdfs:subClassOf gso:SocialConstruct .

default:Crowd
      rdf:type owl:Class ;
      rdfs:subClassOf default:Group .

default:maxArity
      rdf:type owl:FunctionalProperty , owl:DatatypeProperty ;
      rdfs:domain gso:SocialConstruct ;
      rdfs:range xsd:int .

default:Organization
      rdf:type owl:Class ;
      rdfs:subClassOf default:Group .

default:refersTo
      rdf:type owl:FunctionalProperty , owl:ObjectProperty ;
      rdfs:domain default:SocialProperty ;
      rdfs:range gso:Role .
```

## APPENDIX D: CCO IN TURTLE NOTATION

```
@prefix rdfs:    <http://www.w3.org/2000/01/rdf-schema#> .
@prefix sao:     <http://www.openconjurer.org/ontologies/sao.owl#> .
@prefix swrl:    <http://www.w3.org/2003/11/swrl#> .
@prefix protege:  <http://protege.stanford.edu/plugins/owl/protege#> .
@prefix default:  <http://www.openconjurer.org/ontologies/cco.owl#> .
@prefix xsp:     <http://www.owl-ontologies.com/2005/08/07/xsp.owl#> .
@prefix swrlb:   <http://www.w3.org/2003/11/swrlb#> .
@prefix xsd:     <http://www.w3.org/2001/XMLSchema#> .
@prefix owl:     <http://www.w3.org/2002/07/owl#> .
@prefix gso:     <http://www.openconjurer.org/ontologies/gso.owl#> .
@prefix rdf:     <http://www.w3.org/1999/02/22-rdf-syntax-ns#> .

default:Unit
    rdf:type owl:Class ;
    rdfs:subClassOf default:CorporateStructure .

default:isSuperior
    rdf:type owl:InverseFunctionalProperty , owl:ObjectProperty ;
    rdfs:domain sao:Organization ;
    rdfs:range sao:Organization ;
    owl:inverseOf default:isInferior .

default:reportsTo
    rdf:type owl:InverseFunctionalProperty , owl:ObjectProperty ;
    rdfs:domain
            [ rdf:type owl:Class ;
              owl:unionOf (default:Supervisor default:Worker)
            ] ;
    rdfs:range default:Supervisor ;
    owl:inverseOf default:outranks .

default:ManagementBoardMember
    rdf:type owl:Class ;
    rdfs:subClassOf default:Manager .

default:Division
    rdf:type owl:Class ;
    rdfs:subClassOf default:CorporateStructure .

<http://www.openconjurer.org/ontologies/cco.owl>
    rdf:type owl:Ontology ;
    owl:imports                    <http://www.openconjurer.org/ontologies/sao.owl>                    ,
<http://www.openconjurer.org/ontologies/gso.owl> .

default:Supervisor
    rdf:type owl:Class ;
    rdfs:subClassOf gso:Role ;
    owl:disjointWith default:Worker , default:Manager .
```

```
default:Worker
      rdf:type owl:Class ;
      rdfs:subClassOf gso:Role ;
      owl:disjointWith default:Supervisor , default:Manager .

default:isPartOf
      rdf:type owl:ObjectProperty ;
      rdfs:domain gso:HumanAgent ;
      rdfs:range gso:Membership .

default:ManagementBoard
      rdf:type owl:Class ;
      rdfs:subClassOf default:CorporateStructure .

default:Project
      rdf:type owl:Class ;
      rdfs:subClassOf default:WorkingStructure .

default:Manager
      rdf:type owl:Class ;
      rdfs:subClassOf gso:Role ;
      owl:disjointWith default:Supervisor , default:Worker .

default:CorporateStructure
      rdf:type owl:Class ;
      rdfs:subClassOf sao:Organization .

default:isInferior
      rdf:type owl:ObjectProperty ;
      rdfs:domain sao:Organization ;
      rdfs:range sao:Organization ;
      owl:inverseOf default:isSuperior .

default:outranks
      rdf:type owl:ObjectProperty ;
      rdfs:domain default:Supervisor ;
      rdfs:range
              [ rdf:type owl:Class ;
                owl:unionOf (default:Supervisor default:Worker)
              ] ;
      owl:inverseOf default:reportsTo .

default:Workgroup
      rdf:type owl:Class ;
      rdfs:subClassOf default:WorkingStructure .

default:WorkingStructure
      rdf:type owl:Class ;
      rdfs:subClassOf sao:Organization .

default:Team
      rdf:type owl:Class ;
      rdfs:subClassOf default:CorporateStructure .
```