Abstract

The overall concept is that of the Cooperating Smart Space (CSS), which empowers a participant (user or organisation) to interact in one or more Community Interaction Spaces (CIS), to mutually achieve the goals or stated purpose of each Community Interaction Space (CIS). In the process it merges pervasive computing concepts with a more social dimension, for the benefit of participants and 3rd Party service providers.

This document presents the system architecture. It is broken down into three technical sections representing the various viewpoints on the architecture; informational architecture (what information is manipulated), functional architecture (what the system will provide as “core services”) and application architecture (where “core-services” are logically deployed) and related structural elements. Application architecture should be thought of as the “stack” of core services available on each node type.

For third party service providers, the details of the information models and various connection points for third-party services are described in D31. For platform service providers, or platform integrators or operators, the points-of-interoperability i.e. places alternate stakeholders can interoperate with the platform, are described further in D34.

[End of abstract]
Disclaimer

This document contains material, which is the copyright of certain SOCIETIES consortium parties, and may not be reproduced or copied without permission.

In case of Public (PU):
All SOCIETIES consortium parties have agreed to full publication of this document.

In case of Restricted to Programme (PP):
All SOCIETIES consortium parties have agreed to make this document available on request to other framework programme participants.

In case of Restricted to Group (RE):
All SOCIETIES consortium parties have agreed to full publication of this document. However this document is written for being used by <organisation / other project / company etc.> as <a contribution to standardisation / material for consideration in product development etc.>.

In case of Consortium confidential (CO):
The information contained in this document is the proprietary confidential information of the SOCIETIES consortium and may not be disclosed except in accordance with the consortium agreement.

The commercial use of any information contained in this document may require a license from the proprietor of that information.

Neither the SOCIETIES consortium as a whole, nor a certain party of the SOCIETIES consortium warrant that the information contained in this document is capable of use, or that use of the information is free from risk, and accept no liability for loss or damage suffered by any person using this information.

Impressum
[Full project title] Self Orchestrating Community Ambient Intelligence Spaces
[Short project title] SOCIETIES
[Number and title of work-package] WP3
[Document title] D3.3 Revised system architecture
[Editor: Name, company] Micheal Crotty, TSSG
[Work-package leader: Name, company] TSSG
[Estimation of PM spent on the Deliverable] 23 PM

Copyright notice
© 2013 Participants in project SOCIETIES

Optionally list of organisations jointly holding the Copyright on this document
Executive Summary

The purpose of this document is to present an overview of the system architecture. This architecture describes the system at a high level in a technology neutral way. The document starts by defining the major SOCIETIES concepts that need to be understood in order to understand how such a system would operate.

- The Cooperating Smart Space (CSS) concept to represent the various devices that the end-user makes use of and provides a convenient way to manage those devices and the information they contain or generate.

- The Community Interaction Space (CIS) concept which represents a community or group and the information such a group gathers and its life-cycle. It also covers the dynamic nature of membership.

- The Context Model is introduced as a means for capturing the real world situation an end-user or the SOCIETIES system finds itself in. It provides the means to manage this information in a convenient fashion.

- The Preference concept is then introduced as a means to capture the resoluteness and fickleness of an end-user’s choices (for services, for data, or for customisable behaviour). All preference outcomes are assumed to be dependant on the scenario in which they are made.

- The Privacy concepts are presented as the means to capture an end-user’s decisions covering the disclosure of data. It also provides a means for semi-automatically negotiating data access rules based on the end-users previous attitudes to sharing such types of data.

- Finally, the Third party “service” concept is presented as a means of capturing the information necessary, for the system to identity and interact with the third-party service and gauge its capabilities.

The document goes on to cover the functional “Services” that make up the SOCIETIES platform. “Core” services manage the information concepts described earlier, or provide a nice device abstraction or administrative domain abstraction for the SOCIETIES platform. These services are introduced in “groupings” i.e. collections of related services that operate on the same core concept, or operate on the same layer of abstraction.

- CSS Node grouping offers a device abstraction for interacting with a CSS at a basic level.

- CSS grouping offers the services necessary to have a fully functioning CSS for an end-user.

- CIS grouping offers the services required to manage a community, and capture, reason and learn from their collective behaviour.

- Domain grouping presents the infrastructure services required to manage a set of CSS's or CIS's within a single administrative domain.

Finally, the last section describes the view when the system is deployed, and the Node Types and concepts necessary. It defines three Nodes to which the software can be deployed (“cloud” representing any virtualised machine, server or cloud instance), Rich client, a local node capable of operating a CSS and many CIS’s (where latency is a concern), a Light Client node representing a less powerful (mobile) client and a Sensor Node used for capturing real world information, and presenting to the SOCIETIES system. Six Node Types are presented to provide flexibility in the logical deployment of the system and provide functional scalability.

- **Third Party Service Node Type**, to support third-party service providers in provisioning services.

- **Community Node Type** to support provisioning and management one or more CIS’s.

- **Participant Server Node Type** to support one or more CSS’s services in a server environment.

- **Participant Rich Node Type / Participant Light Node Type** to support multiple client capabilities.

- **Domain Node Type**, to provide infrastructure for the management of CSS's and CIS's within a federated administrative domain.
# List of authors

<table>
<thead>
<tr>
<th>Company</th>
<th>Author</th>
</tr>
</thead>
<tbody>
<tr>
<td>HWU</td>
<td>Sarah Gallacher</td>
</tr>
<tr>
<td>IBM</td>
<td>Shiri Kremer-Davidson</td>
</tr>
<tr>
<td>INTEL</td>
<td>David McKitterick</td>
</tr>
<tr>
<td>ITSUD</td>
<td>Z. Wang</td>
</tr>
<tr>
<td>LAKE</td>
<td>Mark Roddy, Liam Marshall</td>
</tr>
<tr>
<td>PTIN</td>
<td>João Miguel Gonçalves</td>
</tr>
<tr>
<td>SINTEF</td>
<td>Jacqueline Floch</td>
</tr>
<tr>
<td>SN</td>
<td>Pierfranco Ferronato</td>
</tr>
<tr>
<td>TRIALOG</td>
<td>Bruno Jean-Bart, Rafik Said-Mansour</td>
</tr>
<tr>
<td>TSSG</td>
<td>Micheal Crotty, Kevin Doolin</td>
</tr>
<tr>
<td>NEC</td>
<td>Sachin Agarwal</td>
</tr>
<tr>
<td>AMITECH</td>
<td>Yiorgas Bouloudis</td>
</tr>
<tr>
<td>ICCS</td>
<td>Ioanna Roussaki</td>
</tr>
</tbody>
</table>
## Table of Contents

1  Introduction ................................................................................................................................. 8
   1.1  Overall vision and concept....................................................................................................... 9
   1.2  Outline.................................................................................................................................... 9
2  Informational Architecture ........................................................................................................... 10
   2.1  Concept CSS .......................................................................................................................... 10
   2.2  Concept CIS ............................................................................................................................ 11
       2.2.1  Conceptual CIS Life-cycle.............................................................................................. 13
       2.2.2  Concept CIS Membership criteria .................................................................................... 14
   2.3  Concept Context Model .......................................................................................................... 15
   2.4  Concept Identity ....................................................................................................................... 17
   2.5  Concept Preference .................................................................................................................. 19
   2.6  Concept Privacy ....................................................................................................................... 20
   2.7  Concept Service ........................................................................................................................ 22
3  Functional architecture ............................................................................................................... 25
   3.1  Domain/Infrastructure grouping ............................................................................................. 25
       3.1.1  CIS Directory .................................................................................................................. 26
       3.1.2  CIS Recommendation ..................................................................................................... 26
       3.1.3  CSS Directory .................................................................................................................. 26
       3.1.4  Domain Authority ........................................................................................................... 26
       3.1.5  Service Marketplace ........................................................................................................ 27
   3.2  CIS grouping .......................................................................................................................... 27
       3.2.1  CIS Information Management .......................................................................................... 27
       3.2.2  CIS Management ............................................................................................................. 27
       3.2.3  Community Context Management ................................................................................. 28
       3.2.4  Community Learning & Reasoning Management ............................................................ 28
       3.2.5  Community Personalisation Management ......................................................................... 28
   3.3  CSS grouping .......................................................................................................................... 29
       3.3.1  CSS Management ............................................................................................................ 29
       3.3.2  Privacy Protection ............................................................................................................. 29
       3.3.3  SN Connector .................................................................................................................. 30
       3.3.4  Service Management ........................................................................................................ 30
       3.3.5  Trust Management & Evaluation ...................................................................................... 31
       3.3.6  User Agent ....................................................................................................................... 31
       3.3.7  User Context Management .............................................................................................. 31
       3.3.8  User Learning & Reasoning Management ......................................................................... 32
       3.3.9  User Personalisation Management .................................................................................... 32
   3.4  CSS Node grouping ............................................................................................................... 32
       3.4.1  Communication Manager ................................................................................................. 33
       3.4.2  Device Manager ............................................................................................................... 33
       3.4.3  Service Discovery ............................................................................................................. 33
4  Application architecture .............................................................................................................. 34
   4.1  Deployment considerations ...................................................................................................... 34
       4.1.1  Functional scalability ........................................................................................................ 34
   4.2  Node Types ............................................................................................................................ 36
       4.2.1  Common functionality ....................................................................................................... 36
       4.2.2  Domain Node Type ......................................................................................................... 37
       4.2.3  Third Party Service Node Type ......................................................................................... 37
       4.2.4  Participant Server Node Type ............................................................................................ 38
       4.2.5  Community Node Type .................................................................................................... 39
       4.2.6  Participant Rich Node Type ............................................................................................... 40
       4.2.7  Participant Light Node Type .............................................................................................. 41
### List of acronyms

<table>
<thead>
<tr>
<th>Acronym</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>C-xx</td>
<td>Community version of the acronym that follows. Eg C-CM = Community Context Management</td>
</tr>
<tr>
<td>CIS</td>
<td>Community Interaction Space</td>
</tr>
<tr>
<td>CSS</td>
<td>Co-operating Smart Space</td>
</tr>
<tr>
<td>CIS-IM</td>
<td>CIS Information Management</td>
</tr>
<tr>
<td>CIS-M</td>
<td>CIS Management service</td>
</tr>
<tr>
<td>CIS-Rec</td>
<td>CIS Recommandation service</td>
</tr>
<tr>
<td>CM</td>
<td>Context Management</td>
</tr>
<tr>
<td>CSSM</td>
<td>CSS Management service</td>
</tr>
<tr>
<td>L&amp;RM</td>
<td>Learning and Reasoning Management</td>
</tr>
<tr>
<td>PrefM</td>
<td>Preference Management service</td>
</tr>
<tr>
<td>PrivM</td>
<td>Privacy Management service</td>
</tr>
<tr>
<td>SN</td>
<td>Social Network</td>
</tr>
<tr>
<td>SNC</td>
<td>Social Network Connector service</td>
</tr>
<tr>
<td>TME</td>
<td>Trust Management and Evaluation</td>
</tr>
<tr>
<td>U-A</td>
<td>(User Agent) User Interaction Monitoring</td>
</tr>
<tr>
<td>U-xx</td>
<td>User version of the acronym that follows. Eg U-CM = User Context Management</td>
</tr>
</tbody>
</table>
1 Introduction

The purpose of this document is to present an overview of the system architecture. This architecture describes the system at a high level, describing the major concepts and data-types manipulated, outlines the “core” services offered, and a logical deployment of those “core services” to meet the requirements specified in requirements deliverables D2.1 and D2.2.

As described in the DoW, this deliverable will “provide the final system architecture for SOCIETIES, providing a comprehensive but high level guidance to the prototype development”. Thus, this architectural deliverable is precluded from making technological recommendations, as it is not intended to provide a specification of prototype implementations, thus this system architecture may be applied to numerous reference implementations.

Readers interested in the technological aspects are encouraged to read the design deliverables which do prescribe suitable technologies, and implementation techniques. Further prototype implementation details can be found as follows:

- D4.1 CSS platform specification and design
- D5.1 Design of Intelligent Community Orchestration Systems
- D5.2 Design of CSS Context Management Systems
- D5.3 Design of CSS Personalisation Systems
- D5.4 Design of CSS Privacy and Trust Systems
- D5.5 Design of CSS User Agent Systems

The inter-operability aspect is of the architecture is specified as a set of “Points-of-interoperability” or access points as defined in the deliverable D3.4. These access points enable the components to be semantically and technologically interoperable. Two types of access points are considered in the deliverable.

- Access points for entities external to the SOCIETIES platform (for instance a 3rd party service). External entities such a legacy system may impose their own constraints, rules, semantic format and their protocols. In such case, Societies shall provide facilities to interact with these different stakeholders regardless their technology.

- Access points between entities of the SOCIETIES Platform. These type of access points are required when these two entities are potentially provided by different stakeholders or are using a communication framework to exchange their information.

The deliverable D3.4 uses a simplified (but compatible) view of the overall architecture in order to highlight those access points.

For the SOCIETIES platform to be very successful, third party developers would have to adapt existing services or provide new services to take advantage of SOCIETIES platform. Deliverable D3.1 describes the “Service model”. It defines the concept of a third party service and its interactions with the rest of the architecture. An important aspect of this work was to provide a prescriptive description of a third party service, in order to highlight how a service can be easily deployed and consumed by the SOCIETIES platform. A simplified version of the “service” concepts are presented here (in D3.3) for completeness, covering the relationship between “core” services and third party services. A more comprehensive, and detailed examination of the informational and functional aspects for third party services is given in D3.1.

This deliverable focuses of describing the capabilities offered by the “Core” services that make up the SOCIETIES platform. Some effort has been made to keep this deliverable short, and refer the reader to other Architecture deliverables as necessary.
1.1 Overall vision and concept

The vision of SOCIETIES is to develop a complete integrated solution via a Co-operating Smart Space (CSS), which extends pervasive systems beyond the individual to dynamic communities of users. CSSs will embrace online community services, such as Social Networking, and thus offer new and powerful ways of working, communicating and socialising.

The goal of SOCIETIES is to radically improve the utility of future Internet services by merging social computing and pervasive computing through the design, implementation and evaluation of an open, scalable service architecture and platform for self-orchestrating Community Interaction Spaces. This will be achieved through four key objectives:

- To facilitate the creation, organisation, management and communication of communities via Community Interaction Spaces (CIS), where pervasive computing is integrated with social computing communities
- To provide an enhanced user experience - both for individuals and entire user communities - based on proactive smart space behaviour and dynamic sharing of community resources across geographic boundaries
- To design and prototype a robust open and scalable system for self-orchestrating Co-operating Smart Spaces (CSS)
- To evaluate, through strong involvement of end-users, the usefulness and acceptance of the developed CSS software via three user trials.

Both major concepts, Community Interaction Space (CIS) and Co-operating Smart Space (CSS), are more formally defined in the Informational architecture section.

1.2 Outline

The document is broken down into three major technical sections, each representing a different view-point of the architecture. The first introduces the information model describing the informational objects and concepts being captured by the system. The second is the functional model which describes what functional features the system will provide. These functional features are specified by a set of “core services” which comprise of the capabilities of the system that can be offered for use by 3rd Party service developers. The final technical section, called the Application Architecture, gives an overview of the logical deployment of the system, and indicates where pieces of functionality reside.

The last section of this document provides a summary and guidance to where to find further design detail on the system specified.
2 Informational Architecture

The following chapter gives a short overview of the major concepts captured and manipulated by the SOCIETIES system.

2.1 Concept CSS

A CSS represents a single participant (user or organisation), and includes their information, and services within a distributed collection of CSS Nodes. It provides both a pervasive capability and a social networking capability in an integrated form. A CSS can interact, communicate, or share directly with another CSS in an adhoc manner.

In order to facilitate interaction between CSS's, and allow the user to manage these interactions in a convenient fashion, the notion of a "contact list" is used. A contact list captures a named relationship between this CSS and other CSS's. For example, a contact list called "friends", capturing CSS's regarded as "friends" by this CSS. This is a unidirectional (directed) relationship, as the other CSS's may not even have a contact list called "friends", and are not required to reciprocate if they did. A Community Interaction Space (CIS) may also be added to a contact list, and the CIS concept is described in the following section.

A key aspect of the following diagram is the use of concepts defined in subsequent sections, to represent the operational data of the CSS. Such operational information includes:

- A set of unique identifiers for the CSS captured as an association to Entity Identifier.
- A model of intent of the participant based on past experience or behaviour of the participant through an association to the UserIntentModel.
- A model of the preferences of the participant, and when these preferences change, captured as an association to Preferences.
- A link to the corresponding context entity linking to the context data (sensor values, profile information etc.) for this participant.

The elements of interest on the diagram and defined in this section are:

CSS - A Cooperating Smart Space (CSS) representing a participant i.e. an end user or an organisation.

CSSNode - It's a logical node on a device or cloud instance running CSS software that coordinates with other CSS Nodes to form a participant's CSS. It has an identity and a transport address as attributes.

This information is captured to provide an administrative boundary on the services operating within the CSS and to support service deployment.

ContactList - This class represents a named set of contacts, for example "friends". The list may contain zero or more identifiers for CSSs. For example, the contact list "family" contains two CSS Identifiers for the CSS of other family members. A Community Interaction Space (CIS) is also identified by an EntityIdentifier thus also can be contained within a Contact List. Finally, a contact list can refer to another included Contact List, to allow a hierarchical organisation. Details of the data representation of the contact list are left for design stage.

CSS Directory - This class represents a conceptual directory of a CSS's public information. It maintains a searchable index of a CSS, and its public (visible) attributes.
2.2 Concept CIS

This section summarises the concept of a CIS, the roles a CSS may take within a CIS, and the logical behaviour of a CIS during its lifecycle. A CIS is a representation of a Pervasive Community and has one or more CSS associated with it. A pervasive community is a group of, two or more, individuals who have agreed to share some, but not necessarily all, of their pervasive resources – personal information, context data, services, devices – with other members of that community.

Members of a pervasive community interact with a CIS via their own personal Co-operating Smart Space (CSS). There is a one-to-one mapping between individuals and CSSs. The only way in which an individual can participate in a CIS is via their CSS but they can also interact with other CSSs without having to form pervasive communities or create CISs.

Individuals may belong to any number of pervasive communities, and thus CISs, simultaneously. Individuals may also, of course, interact with other individuals without using CSSs by employing more traditional mechanisms. The individual members of a pervasive community do not need to be human beings. They can also be organisations, smart space infrastructures, autonomous or semi-autonomous agents, etc. The key defining characteristic of a pervasive community member is the ability to provide and/or take advantage of pervasive technology. Thus, business enterprises, smart shopping malls, robotic companions, etc. can form pervasive communities with human beings or, indeed, with each other.

Pervasive communities can be dynamic in nature, with CISs being formed in an ad hoc fashion as and when required. They can also be created, or become, more permanent and continue to exist even when all the participating members, or their CSSs, are offline. Pervasive communities can also spawn sub-communities or merge with other communities.

The pervasive resources which can be shared via a CIS include, service provision, actuator control and both individual and community context information, preferences and behaviours.

The definition of pervasive community gives us minimum information a CIS should include:

- a unique identity, name and description.
- Membership criteria (can be empty for open/public communities).
- A set of one or more administrating CSSs. Any potential conflicts in administration are resolved via the policy prescribed in the Governship Model.
- Optional set of shared services/resources.
- Optional community centric information such as preferences (via Community Preference Management), intent models (via Community Intent management), context (via Community Context), etc.

This definition defines two relationship roles, one is the role of administration where a CSS may "administrate" a Community Interaction Space (CIS), in the second role a CSS may be a "member of" a given CIS. These two relationships are bidirectional (thus differ from a Contact List introduced earlier). For example, if CSS_A is a member of CIS_work, then CIS_work has a member called CSS_A.

The following diagram depicts the major relationships between a CIS and other related concepts. Concepts prefixed with a name and a "::" are defined in other subsections within this chapter.

![CIS and related information diagram]

The elements of interest defined in this section are:

**CIS** - This is a representation of Pervasive Community. Each CIS has a unique identity, a description outlining briefly its purpose, and a human friendly name.

**CommunityData** - An abstract representation of a community preference, community context, or community user intent.
ActivityFeed – An activity feed object represents activity stream within a CIS. This activity stream is available to all services shared within the CIS, and accessible to all members of a CIS. It contains objects that conform to the subject-verb-predicate structure. For example, “John”, “Published”, “Image23.jpg”

CommunityResource - This represents a reference to a shared resource. This resource is shared with other CSSs that meet the membership criteria of the CIS, to the extent specified in the resource sharing policy. Resources can include services and/or links their corresponding content.

GovernanceModel - This optional aspect covers various mechanisms for deciding or controlling community activity. For example a CIS may have an autocratic model, where a single CSS is asked to decide or control the group.

MembershipCriteria - This is an abstract reference to all membership criteria that can be applied to the CIS. All members must meet these criteria. The criteria allow the membership to be calculated in a dynamic fashion, as needed and on-demand. This allows dynamic open CIS, as only those CSS presenting themselves to a CIS are considered for membership, rather than having a large static list of potential CSS. This adds flexibility to the CIS concept, in scaling to large numbers of participants.

CISDirectory - This class represents a directory of public information available. It maintains a searchable index of a CIS, and its public attributes.

2.2.1 Conceptual CIS Life-cycle

The following diagram summarises the allowable logical states a CIS can assume.

![CIS lifecycle states diagram]

Figure 3: CIS lifecycle states

The above diagram captures the runtime states a CIS may be in. Once a CIS is created and registered, it enters an "Announced" state. In this state, advertisements for the existence of the CIS may be propagated. This announcement is facilitated by the administration CSSs (ie CSSs in an administrative role). The scope of the announcement is controlled by the administrative CSS. For example, for a closed (invite-only) CIS, no announcements may be made, outside of explicit invitations. Also potential members can request what the membership criteria is, and apply to join the CIS.

Once the CIS has one or more members it enters an "Active" state. This is not an automatic transition, and an example illustrates why. The CSS_{ACME} representing ACME company may create a CIS_{new} with criteria only new employees within last six months. Clearly the CSS_{ACME} does not meet these membership criteria, but still can administrate the CIS_{new}. In this “Active” state the shared services of the CIS can be considered as in use by the members. If for any reason the number of active members reaches zero, the CIS returns to an "Announced" state.
A CIS is regarded as “dead” if there are no administrating CSS’s. In such a state the CIS is no longer announced, and serves only as a historical record of the existence of the CIS. It is up to the administration policy of the CIS directory to control how long “dead” CIS’s remain listed. This allows an administrator to re-claim a “deleted” or “dead” CIS.

2.2.2 Concept CIS Membership criteria

In order to support dynamic membership of users in CIS, SOCIETIES provides a generic mechanism for defining rules as membership criteria, e.g.:

- age > 18
- physical distance < 50 meters
- interest = football
- goal = travel to Paris.

Membership is then decided based on whether:

- the user wants to become a member,
- the privacy settings of the user allow the criteria to be evaluated,
- and the criteria are fulfilled.

![Figure 4: CIS membership criteria](image-url)
The major elements of interest are:

**CommunityMembershipRules** - Represents the membership rules of a CIS defined by the CIS administrator(s). It defines generic rules that define which users are invited to join and which users are not allowed to join, based on their profile, context.

**CommunityMembershipOutcome** - Outcome of the application of one instance of CommunityMembershipRules to a CSS.
- ALLOWED - the CIS will accept join requests from this CSS (pending further negotiations...)
- INVITED - the CIS will actively invite the CSS to join it
- NOT_ALLOWED - the CIS will deny join request from this CSS

**UserMembershipRules** - Represents the user rules (or preferences) regarding acceptance into or belonging to a CIS. It defines generic rules that can be processed against each CIS in order to come to a decision regarding the user interest in joining a CIS, based on its profile, context, etc.

**UserMembershipOutcome** - Outcome of the application of one instance of UserMembershipRules to a CIS.
- INTERESTED - the user will attempt to join the CIS
- MAYBE - the user will accept invitations to join this CIS, and the CIS will be considered by the user's CIS recommender
- NOT_INTERESTED - the user will reject any CIS invitations and will not be considered by the user's CIS recommender

### 2.3 Concept Context Model

Context is any information that can be used to characterize the situation of an entity. An entity is a person, place, or object that is considered relevant to the interaction between a user and an application, including the user and applications themselves.\(^1\)

The context management system will model and store the context data collected by the context sources or inserted manually by the users/CSS owners. Some examples of these data are spatial data (location, presence, proximity/vicinity, velocity, acceleration, motion, orientation/direction, address, ...), environmental / physical data (weather, temperature, light, humidity, noise/sound, pressure, weight, size, texture, ...), static and dynamic Personal user data (identity, age, gender, preferences, features, activity, bio parameters, emotional state, social/relationship, ...), temporal information (past/present/future, hours, minutes, seconds, date, day, week, month, season, year, ...). Other types of data that can be considered as context information, such as the properties/status of the available resources (battery life, status (used, owned, suspended, active, installed, ...), capabilities, etc. Additionally, community context, as well as the user/community preferences and intentions will be stored and managed by the context management system.

The following diagram depicts a simplified version of the model used to represent context information.

---

The major elements of interest include:

**Context Model Object** - It is the main class of the context model. All other classes are subclasses of this class.

**ContextEntity** - It is the core concept upon which the context model is built and is further divided to Community and Individual Context Entity. It represents the notion of an Entity that corresponds to an object of the physical or conceptual world. Each ContextEntity contains a “type” property that identifies the nature of the Entity (e.g. PERSON, DEVICE, SERVICE).

**ContextAttribute** - Multiple ContextAttributes can be assigned to a ContextEntity in order to describe its properties. For instance, the NAME, the AGE and the LOCATION of a person would be represented as different ContextAttributes of the person’s ContextEntity.

**ContextAssociation** - It describes relations that may exist among different ContextEntities. Each ContextAssociation is described by a “type” property that escorts it. Associations can be either directed or undirected. The relation described by a directed ContextAssociation exists among non-peer ContextEntity members, one of which is characterized as the “parent”. Example types such ContextAssociations are “locatedIn”, “owns”, “uses”, “teaches”, etc. Undirected ContextAssociations are flat relations among peer ContextEntity members. Example types of such ContextAssociations are: “friends”, “schoolmates”, “fellow passengers”, etc.

**ContextHistoryAttribute** - It is a subclass of ContextAttribute maintaining a subset of the contained information. This class is used in order to store historic context in context history database.

**CommunityMemberContextEntity** - It is a subclass of the ContextEntity class and also a generalization of the CommunityContextEntity and IndividualContextEntity classes.
**IndividualContextEntity** - It is used in order to model individual entities and more specifically a CSS owner (user or organization), which have been grouped formulating a community. It inherits all attributes and operations of the `ContextEntity` class and extends them by methods related to the management of communities the individual belongs to.

**CommunityContextEntity** - It is used in order to capture the context of pervasive communities. Multiple entities forming a community are modelled by this class. It inherits all `ContextEntity` attributes and operations and extends them by methods related to membership management, as well as with Bond attributes that capture the criteria for establishing the community and with context similarity attributes that quantify the similarity among the members’ context. The context model supports the notions of sub-/super-communities, thus allowing for hierarchical communities, as the `CommunityContextEntity` is a subclass of the `CommunityMemberContextEntity`.

**Context Identifier** - This class is used in order to allow the uniquely identification of context model objects. Classes `ContextEntityIdentifier`, `ContextAttributeIdentifier`, `ContextAssociationIdentifier` are subclasses of this class.

**Context Bond** - Common context characteristics shared among community members are described by the `ContextBond` class. Each bond refers to a Context Attribute or Context Association of a particular type that expresses a commonality shared by community members, described by the respective subclasses `CtxAttributeBonds` and `CtxAssociationBonds`.

### 2.4 Concept Identity

There are two interesting basic relationships shown in the following diagram. All identifiers realize the base Identifier contract, as shown at the top of the diagram. This means that all identifiers can be expressed as a URI and have an associated lifetime attribute (i.e. they expire).

The second row of the diagram indicates that an `Entity identifier` is made up of a `Domain Identifier`. Thus an entity is conceptually contained within an Administrative or Identity domain e.g. for example, “ict-societies.eu”. An identity domain is a logical namespace for issuing and managing identities. These are used to break a single global namespace into organisational or administration units, which can be managed in a federated fashion. This means the namespace administration is distributed and not centralised in one entity which may become a bottleneck.

A `Domain Identifier` is globally unique. A `Service Resource Identifier` is made up of an `Entity identifier`, thus conceptually contained within the Entity that offers the service or resource. Each identifier is globally unique by virtue of the composition relationship.

The final rows of the diagram show the type of Entities that have a unique identity. For example, a Service instance has a `Service Resource Identifier`, etc.

---

2 More formally, an identity domain is a subject identifier namespace administered by one entity, typically the digital identity provider. Subjects can authenticate with the administering entity with respect to identifiers that belong to the domain it administers. Subjects that are authenticated with the administering entity inherently trust it to provide claims about other subjects authenticated in the same domain. They may perform the role of relying parties towards the trusted party that authenticates them.
The major elements of interest are:

**DomainIdentifier** – A universally unique identifier for an administrative or identity domain, and the security artefacts related to this identifier. For example, an administrative domain could be "ict-societies.eu" with associated lifetime and certificate attributes. The administrator for an administrative domain is the System Administrator actor identified in the requirements.

**EntityIdentifier** - An identifier for an entity that belongs to a domain, unique for that domain. For example, an entity may be a CSS, a CIS, a 3rd party service provider or a service developer.

**Identity Domain** - An identity domain is controlled by the "System Administrator" actor, and by the Platform Provider. Users have accounts in these domains. One user may have accounts in multiple domains.

**Service Developer** - Identity of the developer of SOCIETIES services. Used for maintaining trust on service installation or update.

**3rd Party Service Provider** - 3rd Party Service Provider who supplies services that use the Societies system.

**ServiceResourceId** - An identifier for a service offered by an entity, unique to that entity. A *ServiceResourceId* offers the greatest granularity in naming individual services or resources.
2.5 Concept Preference

A Preference could be conceived of as an individual's attitude towards a set of objects, typically reflected in an explicit decision-making process. Alternatively, one could interpret the term “preference” to mean evaluative judgment in the sense of liking or disliking an object which is the most typical definition employed in psychology. However, it does not mean that a preference is necessarily stable over time. Preference can be notably modified by decision-making processes, such as choices, even in an unconscious way.\(^3\)

The following diagram summarises the concepts necessary to capture and manipulate dynamically context aware preferences.

![Preference Diagram](image)

**Figure 7: Preference and related concepts**

The major elements of interest are:

**Preference** - this class represents a preference which is a rule that dictates what the user prefers depending on their current context. The Preference is made up of many Preference Node class objects (see below). It also contains other related information such as which service this preference should be used for and which type of services this preference object can be used for.

---

**PreferenceNode** - This class represents a node in the Preference tree. The preference node class can embed a preference condition or a preference outcome. If it embeds a preference condition, then the node is a branch in the tree. If it embeds a preference outcome then the node is a leaf in the tree.

**PreferenceCondition** - The PreferenceCondition class represents a conditional expression such as "location equals home". The condition can be a comparison to a context attribute value, or to a user action represented by a preference outcome or to a user intent task.

**PreferenceOutcome** - The PreferenceOutcome class holds a name - value pair that suggests an action that has to be implemented or a parameter that has to be set. An example is name = "volume" and value = "40".

**PreferenceIdentifier** - The PreferenceIdentifier class represents a unique identifier to the CSS for each preference object.

**UserIntentTask** - The UserIntentTask class represents a list of actions that are commonly repeated in sequence when the user is attempting to achieve some goal.

**UserIntentModel** - The UserIntentModel class represents the entire model of the user's intention based behaviour. It is a hierarchy of UserIntentTasks.

**PreferenceRegistry** - The PreferenceRegistry class represents all the preference identifiers of all the preferences of a CSS. It provides methods that allow a client to retrieve the PreferenceIdentifier of a preference specific to a service or service type.

### 2.6 Concept Privacy

Privacy is defined as the ability of an individual or group to seclude themselves or information about themselves and thereby reveal themselves selectively. Privacy is about the right of an individual to reveal or hide information about themselves to and from other people or organisations. Data collectors are required to state the purpose for which they request the data and the amount of time they will retain the data. This is coined as the "data minimisation" principle derived from Article 6.1(b) and (c) of Directive 95/46/EC and Article 4.1(b) and (c) of Regulation EC (No) 45/2001 which state that personal data must be "collected for specified, explicit and legitimate purposes" and must be "adequate, relevant and not excessive in relation to the purposes for which they are collected and/or further processed" ⁴.

The EU Data Protection Directive 95/46/EC has laid out a number of guidelines with regard to the processing of personal information. A description of the privacy architecture that will attempt to address these guidelines follows. More information on these guidelines and the Data Protection Directive can be found in the home page of the European Data Protection Supervisor⁵.

The platform will attempt to automate privacy decisions based on the user's wishes while allowing the user to control their privacy in every way possible. The architecture comprises of components that implement personalised privacy policy negotiation, use of multiple identities and personalised identity selection and trust management for assessing CSS’s and CIS’s.

A Privacy Policy is a document that describes the policies of a service or a CIS with regards to accessing, processing, storing and sharing personal information. Privacy Policy Negotiation allows the user to negotiate the terms and conditions of a privacy policy for their personal use. The result of a privacy policy negotiation is a privacy agreement that lists the data attributes that the service is allowed to access, the type of actions they are allowed to perform on these data attributes and a number of conditions such as data retention policies, sharing with 3rd parties, the user's right to opt-out of the service etc. Privacy policy negotiation may be seen as benefiting only the user, however, the service can take advantage of offering different level of service based on the amount and quality of information the user agrees to disclose, rather than offering a "one size fits all" approach. It is understandable that this process may be overwhelming to the user who will have to manage this information. The system will attempt to relieve the user from this burden by using context-aware privacy preferences to automate the privacy policy negotiation process.


⁵ European Data Protection Supervisor, http://www.edps.europa.eu/EDPSWEB/edps/site/mySite/pid/74
The use of multiple identities in SOCIETIES allows the user to create different views of their profile to use with different CIS’s and or CSS’s. The user is able to create identities with which they can interact with other CSS’s and CIS’s and reveal different levels and quality of information. The system will be responsible for making sure these identities cannot be linked to the same individual by monitoring the data attributes disclosed with every identity and alert the user of a possible linkage. Personalised identity selection enables the user to set preferences to dictate to the system under what conditions (e.g. specific context situations, when interacting with specific CIS’s or CSS’s) to use each identity.

Trust management enables the system to assess the performance of CIS and CSS’s and use this as input to the privacy policy negotiation process and identity selection process. Feedback from the user about their interaction with services, CSS’s and CIS’s is fed into the trust management components and trust values are assigned to the entities the user interacts with. Trust assessments are also shared with communities and are used as feedback to assess the trustworthiness of CIS’s and CSS’s.

Figure 8: Privacy protection concepts

The major elements of interest are:

**PrivacyAssessment** - Privacy assessment relates to what extend the privacy practices performed on private data are in line with privacy rules (and how the overall privacy policy was respected)

**PrivacyNegotiationAgreement** - This class represents the declarative result (i.e. final negotiation position) of the Privacy Policy negotiation process. It contains the agreement reached between the user and the service provider. It lists all the personal data that the user has agreed to give the provider access to and what operations the provider is allowed to perform on these personal data. It also contains all the obligations the provider and the user have agreed for each item in the personal data such as the data retention period,
sharing with third parties, the right to opt-out etc. The PrivacyNegotiationAgreement document is drafted by the user after a successful negotiation and sent to the provider.

**PrivacyPolicy** - The Privacy Policy is a declarative initial state (ie. initial negotiation position) for the Privacy Policy negotiation process. It is attached to a service provided by an entity (different from the CSS). It. It defines what data this service will request access to / from a CSS personal data and the purpose for which the service is requesting the data. A privacy policy is also attached to a CIS and dictates the type of information a CIS will request access to, the processing practices of the CISs and other conditions such as data retention, storage etc.

The following elements are used to tie these major concepts to the data they protect:

**DataSensitivityMetric** - The Privacy related Data Sensitivity Metric relates each data type to a sensitivity estimate. It provides a measure how sensitive certain data type is for the end – user.

**PrivacyAuditLog** - Privacy log captures different privacy practices that were actually performed on the private data (resources). Privacy practices may cover:

- attempts to read private information
- attempts to modify private information
- attempts to aggregate private information
- attempts to forward private information to other parties
- purposes for what the private information was used

**DataObfuscationPreferences** - This class represents a specialised tree model for data obfuscation preferences.

**IdentitySelectionPreferences** - This class represents a specialised tree model for Identity Selection preferences.

**PPNPreferences** - This class represents a specialised tree model for Privacy Policy Negotiation preferences. There are used to guide or partially automate the negotiation process.

**PrivacyPreferences** - This class is used to represent a privacy preference for privacy policy negotiation, access control and identity selection.

**PrivacyRule** - The Rule encapsulates a Resource, a list of Actions and a list of Conditions. The Rule defines the type of resource the rule applies to using the Resource class, the list of operations that can be performed on the Resource using the Action class and the list of conditions that should be met if access to the resources is to be allowed using the Condition class.

**Action** - The Action class represents an operation that can be performed on a Resource. The Action can be "READ", "WRITE", "CREATE", "DELETE".

**Condition** - The Condition class represents a condition that has to be met by the provider or the user. It is the place where different terms and rights are described: i.e. time retention, share with 3rd Party.

**Resource** - the Resource class is used to represent a piece of data belonging to the user (ie. context data, preference data, profile data). It contains the id of the data and the type of data.

### 2.7 Concept Service

The following diagram depicts the concept of a service within the societies system.

The primary concept in the centre of this diagram is the abstract service class, which has two realisations; a "core" service specified in the functional architecture and a third party service, whose behaviour is developed by an external third party using Societies SDK APIs. All services may use the available APIs of the other "core" services as they see fit.
At the left hand side, a manifest specifies the observable behaviour of the service, in regard to the interfaces of data it offers, or consumes, and provenance information. Details of the (augmented) contents of the manifest for Third Party services are covered in the Deliverable D3.1. The Service Directory allows search of the manifests for given functional services. The service directory is maintained by the Service Marketplace functional service.

Once a service is installed in a CSS, it receives a Service Resource Identifier as shown on right hand side of diagram. It is now a service instance, operating on behalf of the CSS that installed it. The directory of service instances is captured in a Service Registry, which allows the dynamic discovery of transport addresses for the given service instance, and from where they are being offered.

The major elements of interest are:

**Service** - A service is a loosely coupled, reusable software entity that semantically encapsulates discrete functionality and is distributed and accessible programmatically over standard Internet protocols. Loose coupling describes an approach where integration interfaces are developed with minimal assumptions between the sending/receiving parties, thus reducing the risk that a change in one application/module will force a change in another application/module. Services are dynamically discoverable.
Core service - A core service is a service provided by the Societies framework. Any service that is **included as part of the functional architecture** is a "core" service (see the Functional architecture section).

3rd party service - A 3rd party service is a service provided by an external provider. 3rd party services can be hot plugged into a CSS and shared to other CSS, or to members in one or more CIS, according to the service-specific sharing policies defined by the service owner.

Marketplace Directory - This represents a repository of services available for installation. It contains the manifest information for the services as described in D31. These services are publicly offered for installation, i.e. they are available for any CSS to install. The Marketplace Directory is maintained by the Service Marketplace functional service.

Service consumer - Service Consumer is a role played by a component or other software module that requests a service from a Service provider.

Service provider - A service provider is a role played by network-accessible software component that provides a service to any service consumers.

Service registry - Service Registry is a network-accessible registry that accepts and stores *Service Resource Identifiers* (representing an installed service instance) and makes them discoverable to service consumers. The service registry captures the following at runtime:

- available services within this CSS.
- available services shared to this CSS (from another CSS).
- available services shared to this CSS (from available/visible CIS)
- services this CSS shares to other CSS.
- services this CSS shares to other CIS.

The *Entity Identifier* is available indicating the source offering this service instance e.g. whether the service is offered from this CSS, another CSS or a CIS. A service consumer can use this to limit returned results. The service registry concept is maintained by the Service Discovery functional service.
3 Functional architecture

This section provides an overview of the functional architecture for the SOCIETIES project. The following diagram gives an overview of the "core services" provided by the Societies. These are described at a conceptually high level to give an understandable overview of the capabilities of the system.

The services in the diagram are grouped according to the major concept they manipulate or operate on. For example, services that operate on a single CIS are grouped together, as are those that operate on a CSS, and those found on every node in a CSS.

3.1 Domain/Infrastructure grouping

These services operate for the benefit of more than one Community Interaction Space (CIS), or more than one Cooperating Smart Space (CSS) but within one domain. Thus these services operate for a wider open group of stakeholders. They offer federated search and domain administration functions and require multiple CSS or CISs.

This group includes the following services:
3.1.1 CIS Directory

This service is responsible for managing CIS information in a decentralised repository. It uses the CIS concept from the Information architecture. This has the following functions:

- Records available CIS within a domain or set of domains.
- Allows searching for CIS based on specified criteria.
- Allows a CIS to be removed (from the distributed repository).

<table>
<thead>
<tr>
<th>Interface type</th>
<th>Name</th>
</tr>
</thead>
<tbody>
<tr>
<td>Provided:</td>
<td>ICisDirectory, ICisAdvertisementRecord</td>
</tr>
<tr>
<td>Required:</td>
<td>ICisDirectoryCallback</td>
</tr>
</tbody>
</table>

3.1.2 CIS Recommendation

This service is responsible for handling CIS recommendations: recommending a CIS to a user, recommending a user to a CIS. These recommendations will take into account the privacy settings of the users, expressed as an agreed CIS privacy policy. This is also responsible for suggesting the formation of new CIS to a given set of users, given available information.

<table>
<thead>
<tr>
<th>Interface type</th>
<th>Name</th>
</tr>
</thead>
<tbody>
<tr>
<td>Provided:</td>
<td>ICisDataCollector, ICisDataCollectorManager, ICommunitySuggestion, ICommunityPatternAnalysis, ISocialGraph</td>
</tr>
<tr>
<td>Required:</td>
<td>ICisDataSubscriber</td>
</tr>
</tbody>
</table>

3.1.3 CSS Directory

This service is responsible for providing a search capability for CSS’s. The CSS’s are assumed to be managed in a decentralised (federated) registry. This service should allow searching for CSS’s based on public profile attributes and tags. Its duties include:

- Allow discovery of CSS identities, based on specified criteria.

<table>
<thead>
<tr>
<th>Interface type</th>
<th>Name</th>
</tr>
</thead>
<tbody>
<tr>
<td>Provided:</td>
<td>ICssDirectory, ICssAdvertisementRecord</td>
</tr>
<tr>
<td>Required:</td>
<td>ICssDirectoryCallback</td>
</tr>
</tbody>
</table>

3.1.4 Domain Authority

This service represents the provisioning of CSS and CIS identities, and their management. It is assumed to act in a decentralised manner, allowing authentication between multiple domains. This service manipulates the Identity concept. Its duties include:

- Issuing / revoking identifiers and assertions for a CSS.
- Authenticating CSS and CIS to a domain.
- Provisioning of CIS identifiers.
3.1.5 Service Marketplace

This provides access to a repository of installable 3rd Party services and optional “core” services. It includes functionality to query the repository for specific services based on search criteria, to download the corresponding service in a CSS, and provide enabling mechanisms for accounting and charging.

<table>
<thead>
<tr>
<th>Interface type</th>
<th>Name</th>
</tr>
</thead>
<tbody>
<tr>
<td>Provided:</td>
<td>IIdentity, IIdentityManager, INetworkNode, IAuthenticationVerifier</td>
</tr>
<tr>
<td>Required:</td>
<td>IAuthenticationVerifierCallback</td>
</tr>
</tbody>
</table>

3.2 CIS grouping

The following services operate on behalf of a single Community Interaction Space (CIS). A CIS is a representation of a Pervasive Community and has one or more CSS associated with it. There is at least one instance of these services per CIS and an instance of these services can be used by multiple CIS’s.

The services included are:

3.2.1 CIS Information Management

This service is responsible to manage information placed in a CIS by different members. (i.e. CIS content). It manipulates the Information/content objects, and updates a CIS’s context data. Responsibilities include:

- content information storage, and retrieval,
- optional version control,
- sharing rules expressed as a privacy policy,
- recommendations on CIS information.

<table>
<thead>
<tr>
<th>Interface type</th>
<th>Name</th>
</tr>
</thead>
<tbody>
<tr>
<td>Provided:</td>
<td>IActivity, IActivityFeed, IActivityFeedManager, ICommunityData, ICommunityDataVersioning, ISuggestedCommmunityData</td>
</tr>
<tr>
<td>Required:</td>
<td>IActivityFeedCallback, ISuggestedCommunityDataCallback, ICommunityDataEventListener</td>
</tr>
</tbody>
</table>

3.2.2 CIS Management

This service is responsible to handle all aspects of CIS lifecycle management. Each instance manages zero or more CISs. It manipulates the CIS, CIS Membership, and CIS Governance concepts from the Information architecture. Its capabilities include:

1. CIS profile manager (available and/or visible attributes etc.)
2. Membership manager (who is a member?)
3. Roles manager (what roles are supported and who does what?)
4. Lifecycle operations (e.g. CIS creation)

<table>
<thead>
<tr>
<th>Interface type</th>
<th>Name</th>
</tr>
</thead>
<tbody>
<tr>
<td>Provided:</td>
<td>ICis, ICisParticipant, ICisAdministrator, ICisManager, ICisMembershipCriteria</td>
</tr>
<tr>
<td>Required:</td>
<td>IManagerCallback</td>
</tr>
</tbody>
</table>

### 3.2.3 Community Context Management

This service represents access to and maintenance of "community" context. For each CSS there is one instance of community context management responsible for all CISs administered by this CSS maintaining the current "community" context. This service instance may manage more than one "community" context. It manipulates the context model. Its duties include:

- Allowing query/update operations for "community" context.
- Allowing addition/update/removal operations for "community" context.
- Maintaining the history of "community" context for a CIS.
- Interact with community learning and reasoning for inferred or deduced context information.

<table>
<thead>
<tr>
<th>Interface type</th>
<th>Name</th>
</tr>
</thead>
<tbody>
<tr>
<td>Provided:</td>
<td>ICTxBroker, ICTxCtxEntity, ICTxMemberCtxEntity, ICTxBond, ICTxAssociationBond, ICTxCtxHistoryMgr</td>
</tr>
<tr>
<td>Required:</td>
<td>ICTxEventListener, ICTxChangeEventListener, ICTxAttribute, ICTxAttributeValue, ICTxIdentifier, ICTxModelObject, ICTxAssociation, ICTxEvent</td>
</tr>
</tbody>
</table>

### 3.2.4 Community Learning & Reasoning Management

This service offers functionality that allows learning community behaviour models, such as community preferences and community intent using the activity history of its community members. This service operates based on the Context Model from the Informational architecture.

<table>
<thead>
<tr>
<th>Interface type</th>
<th>Name</th>
</tr>
</thead>
<tbody>
<tr>
<td>Provided interfaces:</td>
<td>ICTxCtxInferenceMgr, ICTxCtxInheritanceMgr, ICTxCtxPredictionMgr, ICTxCtxRefiner, ICTxCtxSimilarityEvaluator</td>
</tr>
<tr>
<td>Required interfaces:</td>
<td>ICTxEventListener</td>
</tr>
</tbody>
</table>

### 3.2.5 Community Personalisation Management

The Community Personalisation Management system manages (stores, retrieves) the community preferences and community intent and exposes interfaces for community members to retrieve these preferences and intent models for their own use. It manipulates the Preference and User intent concepts in the information model.
### 3.3 CSS grouping

These services operate on behalf of a single participant. A participant is used to *refer to a single user or organisation*. There is at least one instance of these services per participant and an instance of these services can be used by multiple participants.

It includes the following services:

#### 3.3.1 CSS Management

This manages activities associated with a CSS. It covers which CSS Nodes (devices or cloud instances) are part of the CSS, and assigns a common identity. A CSS represents a single participant. This service also looks after configuration of shared resources i.e. the definition of sharing policies or rules. For example, how much of a service is shared with a specified CIS. Its duties include:

- CSS Node provision and lifecycle management, including the synchronisation of CSS Management data between CSS Nodes.
- Initial CSS provision and lifecycle management
- Managing resource sharing and configuration policies (but not enforcement, or conflict resolution)

#### 3.3.2 Privacy Protection

This service includes mechanisms for the management of the privacy policies to be used by a CSS. This privacy policy will describe the terms and conditions the CSS will respect concerning the personal data. The default policy will be strict enough to avoid any personal data to be disclosed without the user consent.

- To help the user, Privacy Preferences facilities will be available to manage personal privacy settings.
- The service will include Privacy Negotiation facilities to manage the dynamic aspect of communities where a user can negotiate privacy policies with service providers.
- An Identity Management will provide a mechanism to enable the user to use the Domain Authority capabilities and select the right level of identity in each CSS he will participate.
- Risk & Trust Management facilities will provide a means to warn the user about the risk to participate in a community and manage the trust level a user may have about CIS members or services.
### 3.3.3 SN Connector

This service represents the mechanism to interact with an existing Social Network (SN). The SN connector will be built around federated social networks. This allows extraction of public info available in the SN, as well as access/updating of non-public information for the specified user.

Proprietary social networks can also be accessed via their proprietary APIs (for example, FaceBook's Graph API). This will allow the extraction of information out of existing (non-federated) social networks, but not updating. These aspects will be modelled as context sources.

Key duties of this service include:

- authentication to SN (login, logout).
- public query functionality (access public profile information of other users)
- query functionality (profile information for authenticated user)
- query social graphs ("Friend's" or "Collections" of contacts)
- publishing content to a Federated SN

### 3.3.4 Service Management

Service Management refers to the setup and lifecycle control of a 3rd Party service or resource in correlation to a CSS. Further details on the contractual requirements on 3rd Party services can be found in D3.1 and are not covered here. Service provisioning covers the following capabilities:

- Installation of new 3rd party services.
- Configuration/reconfiguration of 3rd party services.
- Enforcement of sharing policies for 3rd party services, when access is required.
- Removal of 3rd party services
3.3.5 Trust Management & Evaluation

The Trust Management & Evaluation (TME) service is responsible for collecting, maintaining and managing all information that is required when making trust-based decisions. It includes a Trust Engine for evaluating direct, indirect and user perceived trust. Every party, i.e. user, community or third party service, is evaluated periodically with regards to its trustworthiness, based on various criteria, by aggregating experiences from direct interactions and trust recommendations from other trusted entities. Thus, the Trust Engine monitors the trust level of the entities over time by applying composite evaluation mechanisms and it provides information for the level of each party's trustworthiness in a highly dynamic environment. Finally, access to the assessed trust values is enabled by the TME service.

<table>
<thead>
<tr>
<th>Interface type</th>
<th>Name</th>
</tr>
</thead>
<tbody>
<tr>
<td>Provided:</td>
<td>ITrustBroker, ITrustEvidenceCollector, TrustEvent,</td>
</tr>
<tr>
<td></td>
<td>TrustValueType, TrustedEntityType, TrustedEntityId</td>
</tr>
<tr>
<td>Required:</td>
<td>ITrustEventListener</td>
</tr>
</tbody>
</table>

3.3.6 User Agent

The User Agent acts on behalf of a single CSS. It gathers information from several components in the system, such as preferences and user intent from the User Personalisation Management System, context information from the Context Management System, service status information from the Service Management System etc.

Based on these input sources, services are provided in a proactive manner, conflicts are resolved and decisions are made on behalf of the user.

Meanwhile user involvement is supported during the decision making process, and the User Agent also provides the appropriate mechanisms for feedback.

<table>
<thead>
<tr>
<th>Interface type</th>
<th>Name</th>
</tr>
</thead>
<tbody>
<tr>
<td>Provided:</td>
<td>IUserActionMonitor, IResourceSharing, Isharable,</td>
</tr>
<tr>
<td></td>
<td>IUserFeedback, IConflictResolutionManager,</td>
</tr>
<tr>
<td></td>
<td>IDecisionMaker</td>
</tr>
<tr>
<td>Required:</td>
<td>IAction, IActionConsumer</td>
</tr>
</tbody>
</table>

3.3.7 User Context Management

This service includes a variety of functionality related with the management of the user's context information. The service is responsible for acquiring this information from sensors and other context sources, for modelling the collected data and for maintenance of current and historic context information in appropriate data repositories. Additionally, inference techniques are provided enabling the extraction of high level information from raw context data. Management of context information also includes the provision of interfaces to context consumers enabling them to retrieve, update or delete stored context data. Finally, propagation of context information and synchronisation of the various context repositories is also supported.
3.3.8 User Learning & Reasoning Management

The User Learning & Reasoning Management system includes all the components responsible for learning user behaviour models such as user preferences, user intent models, Bayesian models etc using the user's history of actions stored in the system. This includes more general algorithms as well as specialised learning and reasoning algorithms. These specialised specific algorithms offered by this system are used to provide prediction models that are processed by the User agent system, preference models that are processed further by the Preference management system.

<table>
<thead>
<tr>
<th>Interface type</th>
<th>Name</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Provided:</strong></td>
<td>ICtxBroker, ICtxSourceManager, IndividualCtxEntity, ICtxAttribute, IctxAttributeType, ICtxAttributeValue, ICtxIdentifier, ICtxModelObject, ICtxAssociation, ICtxAssociationIdentifier, ICtxEvent, IctxChangeEvent, IctxQuality, IUserCtxHistoryMgr</td>
</tr>
<tr>
<td><strong>Required:</strong></td>
<td>ICtxEventListener, ICtxChangeEvent</td>
</tr>
</tbody>
</table>

3.3.9 User Personalisation Management

The User Personalisation Management component manages the user behavioural models such as the user preferences, user intent, Bayesian models and others. Its functionality includes the evaluation of these models, monitoring of the context of the user and predicting the user's actions. As a result, it will suggest actions to be taken by the User Agent service.

<table>
<thead>
<tr>
<th>Interface type</th>
<th>Name</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Provided:</strong></td>
<td>IUserLocation, Icoordinate, IuserCtxInferenceMgr, IuserCtxInheritanceMgr, IuserCtxPredictionMgr, IuserCtxRefiner, IUserCtxSimilarityEvaluator</td>
</tr>
<tr>
<td><strong>Required:</strong></td>
<td>ICtxEventListener</td>
</tr>
</tbody>
</table>

3.4 CSS Node grouping

The core services in this group are available per CSS Node. A CSS Node is a logical node/device/cloud instance running CSS software that coordinates with other CSS Nodes to form a participant's CSS. There is an instance of these services per CSS Node.

This grouping includes:
3.4.1 Communication Manager

This service provides the necessary mechanisms for managing any communications within the CSS and with a CSS Node of other CIS members. This will cover network management responsible for identification and maintenance of network connections, discovery of CSS Nodes (devices), communications between discovered nodes and the switching between the local and cloud modes. It will manage high level tasks such as:

- detecting networks, setting up connections, monitoring connection status and switching connections to other network types when better networks available,
- providing identity based messaging and routing for both inter and intra CSS messages,
- message prioritisation and delivery mechanism selection unicast or multicast.
- notification of failed messaging attempts.

<table>
<thead>
<tr>
<th>Interface type</th>
<th>Name</th>
</tr>
</thead>
<tbody>
<tr>
<td>Provided:</td>
<td>ICommManager, IPubsub, IEventManager, ICommManagerFactory</td>
</tr>
<tr>
<td>Required:</td>
<td>ICommCallback, ISubscriber, IEventListener</td>
</tr>
</tbody>
</table>

3.4.2 Device Manager

This service provides the mechanisms for managing devices within a CSS. It covers the process for discovering hardware devices and the management of the capabilities of those devices (for example, available hardware sensors, and corresponding context sources).

It optionally provides context sources for device/cloud instance specific attributes, for example, available memory, processing power, etc. These context sources are managed a part of user context management.

<table>
<thead>
<tr>
<th>Interface type</th>
<th>Name</th>
</tr>
</thead>
<tbody>
<tr>
<td>Provided:</td>
<td>IDevice, IAction, IDriverService, IDeviceStateVariable</td>
</tr>
<tr>
<td>Required:</td>
<td>IEventListener</td>
</tr>
</tbody>
</table>

3.4.3 Service Discovery

This service provides service discovery and advertisement features. It provides mechanisms for finding services. This allows discovery of core platform services within this CSS, and 3rd party services within available CIS or other CSSs at this point.

<table>
<thead>
<tr>
<th>Interface type</th>
<th>Name</th>
</tr>
</thead>
<tbody>
<tr>
<td>Provided:</td>
<td>IServiceDiscovery</td>
</tr>
<tr>
<td>Required:</td>
<td>IServiceDiscoveryCallback</td>
</tr>
</tbody>
</table>
4 Application architecture

The application architecture is the architecture view-point concerned with the structural and deployable units required to build a component-based system that conforms to the functional and non-functional requirements. It addresses the set of conceptual structures and groupings need to build a system that can be flexibly deployed and support a wide range of scenarios with varying scalability requirements.

4.1 Deployment considerations

The following section summarises how the different node types defined could be deployed to real nodes.

A node is a physical device or virtual machine instance, which is capable of executing deployed services. It also gives a mapping between the Node Types and corresponding Nodes. For our purposes:

A node-type is a logic grouping of deployable services/components that provide a well defined set of functionality, when deployed to a Node. More formally, a Node type is a set of software components with related technologies that provide a set of well defined functionality given via interfaces and usage patterns.

Components/services in another Node Type can interact and execute services without being aware of the implementation details. Node Types are deployable units that offer well specified functionality that can be replicated if needed on another Node.

4.1.1 Functional scalability

The SOCIETIES core platform is split to allow flexibility in deployment for Participant, Community and Domain entities. This split manifests itself Nodes and Node Types. For example, a conference venue may want to host as many CISs as possible, and will require more than one instance of the Community Node Type to do so. It is unlikely that a conference provides personal CSS to all participants so one instance of the Participant Node type is all that is required to provide a “Conference venue” CSS. A conference venue also needs a single instance of the Domain Node type, to have control over issuing CIS identifiers, and a single instance of the Third party Service Node Type to host Third party (Conference specific) services. Thus splitting the SOCIETIES platform provides deployment flexibility, and some degree of functional scalability.

A second aspect is the provision for some limited data-partitioning for CSS, CIS and Administrative domains. For example, a CIS is allowed to have one or more administrations. Clever selection of the community “administrators” for a large group allows community administration activities to be load-balanced across administrators and thus Nodes. Any potential conflicts are resolved using the specified Governance Model.

Also each CSS logically belong to a Identity or Administrative domain (see Domain NodeType) As these Administrative domains operate in a federated fashion, it can be advantageous to split the large user group into a number of smaller Administrative domains.
Figure 11: Mapping Node Types to deployment Nodes

The above diagram summarises the deployment options to a typical deployment environment envisioned by SOCIETIES. The nodes of interest are:

**Cloud Node** - This node represents a deployment to a server based device or cloud instance. As the execution environment is very similar in cloud based and server based nodes, no distinction is drawn between these environments.

It runs a server based OS, for example, Ubuntu server, Windows Server, or OpenSolaris, etc. The exact choice of OS will be made as part of the design decisions in Task 4.2, Device and communication abstractions.

**Rich Client Node** - This node represents a deployment to standard OS. This environment is deemed "rich" in the aspect that the processor, memory, and available disk space is greater than available in a "Smart" Participant Device.

Examples include Linux, Windows, or Mac OSX. The exact technological choice will be made as part of the design decisions of Task 4.2. (Device and Communication Abstractions)

**Sensor Node** - This node represents a device running an embedded OS, or pre-configured firmware that allows it to perform sensing functions. It is assumed capable of communicating via some protocol with a SOCIETIES node. It is included here for completeness, but does not run SOCIETIES services.
**Light Client Node** - This node represents a deployment to a smart device running a mobile OS. Examples include Android.

### 4.2 Node Types

This section summarises the Node Types envisioned by the SOCIETIES system as shown in the following diagram. Common Functionality is an abstraction of the commonality between all other node types, and represents the common services that are in all other platform types.

![Node Type Hierarchy Diagram](image)

*Figure 12: Node type hierarchy*

These Node Types are described in subsequent sections.

#### 4.2.1 Common functionality

**Common functionality** defines the mandatory services required to be supported on all CSS Nodes, in order to be able to effectively communicate with other node instances. These core services are:

- Communication Manager providing an abstraction on the messaging and network infrastructure to send and receive messages.

- Device Manager providing the means to detect attached sensors and devices to the CSSNode, and register them if appropriate as Context Sources.

- Service Discovery providing the ability to find core services within this CSS, as well as third party services in this CSS, other services available via other CSSs, and services available via active CIS.
4.2.2 Domain Node Type

This node type runs on a cloud node and contains the services necessary to support a specific administrative domain. In almost all cases, there is need for only one instance of this node type in an (business or personal) administrative domain.

A domain has one instance of the following services:

- Domain authority, containing a CSS registry and authentication and search functionality.
- CIS registry containing a set of available CISs within this domain.
- These services include CIS Recommendation. This service is included as it provides recommendations for users to join CIS, or form new CIS.

4.2.3 Third Party Service Node Type

The node type represents an integration point for Third party applications and services. It supports a Third party service runtime. No further obligations are made on this node type, other than those outlined in the Common Platform. This gives substantial flexibility in the implementation of third party services.
4.2.4 Participant Server Node Type

This node type represents the core services that operate on behalf of a participant running on a cloud or server node. This node type provides complex learning and reasoning algorithms to operate on this data e.g. User Learning and Reasoning service. These core services are placed into this node type as they require:

- access to a large volume of participant data,
- or need to run for an extended period,
- or be always available.

Figure 15: Additional Core services in Third Party Node Type

Figure 16: Core services in Participant Server Node Type
4.2.5 Community Node Type

This Node Type contains all the services necessary to support the creation, management and maintenance of a CIS. In other words these services act on behalf of a CIS to provide benefit for all members of a CIS. These core services can be deployed to a cloud or server node and are placed into this platform as they require:

- access to a large volume of community data,
- or need to run for an extended period,
- or be always available to CIS members i.e. other member CSS.

![Figure 17: Core services in Community Node Type](image)

Figure 17: Core services in Community Node Type
4.2.6 Participant Rich Node Type

This client platform represents the services and components necessary to allow a user to interact with the SOCIETIES system in a feature rich way. It interacts as a client to all other Node Types, for example, for authentication the domain Node Type, for Community Context the community Node Type, etc.. It contains the core services required, and management GUIs needed to operate on a desktop computer, laptop, interactive display, or tablet PC. In it envisioned that this platform has the ability to provide limited third party services tailored to the environment they operate in.

Figure 18: Core services in Participant Rich Node Type
4.2.7 Participant Light Node Type

This client platform represents the services and components necessary to allow a user to interact with the SOCIETIES system. More specifically, it allows interaction with the Community, Participant Server. It contains the minimum core services required, and corresponding management GUIs. It is designed to operate on a smart phone or lower specification tablet PC. Thus it is envisioned that these components:

- are customised to the mobile OS on which they run,
- have limited battery life,
- are portable enough to be carried with the end-user.

![Diagram of Participant Light Node Type]

Figure 19: Core services in the Participant Light Node Type
5 Summary

The following diagram provides a summary of the system architecture.

![System overview diagram]

Legend: U- = User variant, C- = Community variant

- CM: Context Management
- CIS-IM: CIS Information Management
- CIS-M: CIS Management service
- CIS-Rec: CIS Recommandation service
- L&RM: Learning and Reasoning Management
- PrefM: Preference Management service
- 3p Service GUI
- Platform GUI
- 3p Service Node Type
- Participant Node Type
- Community Node Type
- Runtime env
- Participant Rich Node Type
- Participant Light Node Type
- Service Provisioning
- Domain Authority
- CIS Recommendation
- CIS Directory
- CSS Directory
- Message Routing
- Domain Node Type
- Federated Social Network
- Legacy Social Network
- Facebook / LinkedIn
- 3p Service App
- Platform GUI
- CSSM
- Proxies
- SD
- Comm
- Mobile OS
- Other federated instances
- Decision maker, and monitor
- User Agent
- Trust Management and Evaluation
- Social Network Connector service
- Privacy Management

Figure 20: System overview
The previous diagram maps the functional services to the corresponding Node Types. This gives a overview of the deployment potential for the system as a whole. The Domain Node Type acts as an infrastructure hub for the other Node Types to interact with. It also federates with other administrative/identity domains to ensure different deployments can interoperate.

The Community Node Type offers all the service necessary to establish and manage a set of CIS's. This includes the ability to learn, infer, predict, and estimate community context, community preferences and community intent.

The Participant server Node Type, allows a good deal of the processing required to support a CSS to be off-loaded to a server environment (in the cloud or otherwise). The services deployed can support one or more CSS's, benefit from “always-on” access and store and process large amounts of historic data, that will enable the platform to learn in an accurate and beneficial way.

The Service Node type allows the system to give a dedicated service provisioning environment to Third party service developers. Additional nodes can be provisioned if some third-party services are incompatible with each other when co-hosted, require additional security and reliability ie. redundant nodes, or just need to functionally scale to meet end-user demand.

The Participant Rich Node client is used to provide a highly functional client within the local environment. This is useful where there are bandwidth issues, latency or data sensitivity concerns when using the cloud nodes. It contains functionality for hosting a CSS or hosting one or more CIS's, and performing learning activities locally.

The Participant Light Node Type contains those services necessary to allow and end-user to interact with the SOCIETIES platform in a meaningful way. Some of these components are light or “proxy” versions where a small implementation exists on the device, but the majority of the associated processing is forwarded onto the cloud node. This also has the advantage of reducing the amount of processing on the local device thus conserving battery usage.

Combined the “core” services these Node Types form the basis of Service-oriented Architecture (SOA) that makes up the SOCIETIES platform.

### 5.1 Further details

This document represents an overview of the system architecture. Further details for various aspects can be found in:

- **D31 Service Model Architecture.** This deliverable describes the system from a Third Party service providers view point. It highlights the details of service description, and service partitioning so that a Third Party service can be deployed on the various Node Types.

- **D34 Interoperability Architecture.** This deliverable describes the “Points of interoperability” or access points of the SOCIETIES platform as a whole. This deliverable covers the various types of interoperability from semantic to technical. It also details various access points (where additional functionality can be plugged in), as well as external system integration points e.g. the connection to existing social network providers, or the integration points with the Android OS.

Further design and implementation details can be found in design deliverables as follows:

- **D4.1 CSS platform specification and design**
- **D5.1 Design of Intelligent Community Orchestration Systems**
- **D5.2 Design of CSS Context Management Systems**
- **D5.3 Design of CSS Personalisation Systems**
- **D5.4 Design of CSS Privacy and Trust Systems**
- **D5.5 Design of CSS User Agent Systems**
To aid quick comprehension by third party developers the prototype implementation uses a simplified architecture view as follows:

**SOCIETIES Architecture**

In this view some of technological prototype implementation details have been included, for example, XMPP messaging, or Android deployment. In addition some of the Node Types that make up the platform are combined for the sake of clarity as this view focuses only on the deployment nodes.

The minor differences in the architecture views are to highlight the more important aspects for the various stakeholders; third-party service providers who make use of the SOCIETIES platform or SOCIETIES Platform providers/resellers who may extend the “core” services in the platform.
6 Glossary

**ACTUATOR**

An actuator is a mechanical device for moving or controlling a mechanism or system.
[from Wikipedia]

**ADMINISTERING CSS**

This is a role a CSS can adopt. It does not have to meet the membership criteria of the CIS and it has the following duties.
- To ensure the CIS is registered/advertised to other CSSs.
- To enforce the membership criteria (when new participant CSS request to join the CSS).
- To broker a governance model, where decisions are sent to the most appropriate CSS(s) to handle it. For example, in a democratic model, all other members vote on a decision, the administrating CSS collects these votes, and informs the requester(s) of the decision. The administrating node does not make the decision just facilitates the decision making process.

**ARCHITECTURE VIEWPOINT**

It is a technique for abstraction using a specific set of architectural concepts and structuring patterns, in order to focus on particular concerns within a system.

**ASSOCIATION**

A relationship between two or more entities. Implies a connection of some type - for example one entity uses the services of another, or one entity is connected to another over a network link.

**BROKER**

A business role: Individuals or companies that act as intermediary between two parties. For example, brokers who exploit data discovered by SOCIETIES platform for the use of 3rd party services.

**CIS**

see Community Interaction Space

**CIS SERVICE**

A service that is offered as part of a CIS's capabilities, where the access is provided using a prescribed graphical user interface and/or API.

**CSS**

see Cooperating Smart Space

**CSS NODE**

It's a logical node/device/cloud instance running CSS software, that coordinates with other CSS Nodes to form a participant's CSS.

**CLASS**

A logical entity encapsulating data and behaviour. A class is a template for an object - the class is the design, the object the runtime instance.

**COMMUNITY**

It's used to describe the collection of participants with common interests or purpose. It is defined as follows:
"a social, religious, occupational, or other group, sharing common characteristics or interests and perceived or perceiving itself as distinct in some respect from the larger society"

**Reference:**

**COMMUNITY CONTEXT**

A set of context information derived from a group of end-users or their devices in a specific Community. This data set is dynamically formed from in-range CSSs, individual privacy settings,
filtered and combined using conflict resolution techniques and context inheritance mechanisms. (see also Context)

**COMMUNITY INTERACTION SPACE**

It's a representation of a Pervasive Community and has one or more CSS associated with it. It includes:

- a unique identity, name and description.
- membership criteria (can be empty for open/public communities)
- a set of one or more administrating CSSs.
- a dynamic membership list of member CSSs.
- a set of shared services/resources.
- optional community centric information such as preferences, intent models, context, etc.

**COMMUNITY ORCHESTRATION**

The ability to help users manage the intelligent identification, formation, organisation, membership and termination of communities. This ability is based on end-user supplied rules for community life-cycle management and community membership and associated context information.

**COMMUNITY PREFERENCES**

A stereotypical set of preferences or template, for a whole community, based on the preferences of individual community members. (see also Preference)

**COMMUNITY PROVIDERS**

A business role: These are stakeholders that specialize in CIS management including the customization and the definition of communities for specific customer needs. This stakeholder is not involved in every community definition but yet is needed when its client business or organization does not have the required technical skills or will to do this on their own. Example stakeholders: community discovery providers and community creators.

**COMMUNITY ADMINISTRATION**

Community administration is the procedures and actions performed in order to maintain, arbitrate and orchestrate the participants of the community in terms of the hardware and software components.

**COMMUNITY CREATION CRITERIA**

The judgement or decision that supports the creation of a dynamic or not community, based on the initial characteristics and the requirements of the user/group/event that triggers the formulation of a community.

**COMPONENT**

A business component is the software implementation of an autonomous concept or process. It consists of all the software artefacts necessary to represent, implement, and deploy a given business concept as an autonomous, reusable element of a larger distributed information system.

**COMPONENT MODEL**

The component model provides a detailed view of the various hardware and software components that make up the proposed system. It shows both where these components reside and how they inter-relate with other components. Component requirements detail what responsibilities a component has to supply functionality or behavior within the system.

**COMPUTER SUPPORTED CO-OPERATIVE WORK**

Computer supported co-operative work (CSCW) addresses "how collaborative activities and their coordination can be supported by means of computer systems".

**Reference:**


CONFIDENCE

Confidence is generally described as a state of being certain either that a hypothesis or prediction is correct or that a chosen course of action is the best or most effective.

Reference:
World Database of Trust Harvey S. James, Jr., Ph.D (Updated August 2007) A variety of definitions of trust are collected and listed.

CONFIDENTIALITY

Confidentiality in a general sense refers to the duty not to share information with persons who are not qualified to receive that information. In a more specific sense, it refers to the confidentiality of communications provided for in Article 5 of the E-privacy Directive 2002/58/EC and in Article 36 of Regulation (EC) No 45/2001.
Confidentiality of processing also refers to the obligation of any person acting under the authority of the controller or the processor, who has access to personal data, not to process them except on instructions from the controller, unless he is required to do so by law (see Article 16 of Directive 95/46/EC and Article 21 of Regulation (EC) No 45/2001).

Reference:

CONFLICT RESOLUTION

Conflict resolution is a range of methods of eliminating sources of conflict. Conflict is actual or perceived opposition of needs, values and interests. A conflict can be internal (within oneself) to individuals. Conflict, as a concept, can help explain many aspects of social life such as social disagreement, conflicts of interests, and fights between individuals, groups, or organizations.

Reference:

CONSENT

In data protection terminology, consent refers to any freely given, specific and informed indication of the wishes of a data subject, by which he/she agrees to personal data relating to him/her being processed (see Article 2 sub (h) of Data Protection Directive 95/46/EC and Article 2 sub (h) of Regulation (EC) No 45/2001.
Consent is an important element in data protection legislation, as it is one of the conditions that can legitimise processing of personal data. If it is relied upon, the data subject must unambiguously have given his/her consent to a specific processing operation, of which he/she shall have been properly informed. The obtained consent can only be used for the specific processing operation for which it was collected, and may in principle be withdrawn without retroactive effect.

Reference:

CONTENT PRODUCERS

A business role: Individuals or companies that develop content e.g. Films, music, games, weather forecast, news, advertisements, etc.

CONTENT PROVIDERS

A business role: These are companies that make content available via community services to targeted end-users. They obtain these distribution rights from Content Producers. Examples include: Conference Organizers, University Administrators.

CONTEXT

Context is any information that can be used to characterize the situation of an entity. An entity is a person, place, or object that is considered relevant to the interaction between a user and an application, including the user and applications themselves.

Reference:
CONTEXT CHANGE

An alteration in an individual's or a community's context information. This alteration can be the addition of new information, the removal of old information, or an update of existing information. (see also Context)

CONTEXT PROVIDERS

A business role: Companies that can provide context information (e.g. location) that could be used as input to services implemented on top of the SOCIETIES platform. These companies could also act as combiners for various existing context sources.

CONTEXT STATE MODEL

A context state model is a filtered view of a community's or an individual's context information used for analysing community membership. Context State Models are formed using trajectory data mining techniques, to give a snapshot of pertinent context information for stereotypical community memberships. (see application of technique in Community Orchestration)

CONTEXT AWARENESS

Context-awareness refers to the idea that computers can both sense, and react based on their environment. Devices may have information about the circumstances under which they are able to operate and based on rules, or an intelligent stimulus, react accordingly. Context-aware devices may also try to make assumptions about the user's current situation. In addition, context-aware systems are concerned with the acquisition of context (e.g. using sensors to perceive a situation), the abstraction and understanding of context (e.g. matching a perceived sensory stimulus to a context), and application behaviour based on the recognised context (e.g. triggering actions based on context).

Reference:

CONTEXT INFERENCE

Context inference (also known as context reasoning) is the process of extracting the probability that a particle of contextual information would be in a specific state after some kind of interaction with other context information or the environment. It is the process of making context information explicitly available from other context sources (called lower-level context in this regard). It is part of context refinement and is based on drawing conclusions from existing information.

Reference:

CONTEXT MANAGEMENT

Context management includes the distributed context maintenance, access, update and synchronisation, as well as, the provision of a transparent interface to context handling, support of ad-hoc context exchange, real-time and non-real-time context handling.

Reference:

CONTEXT OF USE

Context of use for a product: the goals of the user community, and the main user, task and environmental characteristics of the situation in which it will be operated.

Reference:

CONTEXT REASONING

See Context refinement
CONTEXT REFINEMENT
Context refinement can be any method that accesses available context information of any kind and refines/enriches it, with the right quality and relevance. It is any process that creates new knowledge based on available context information.

Reference:

COOPERATING SMART SPACE
A CSS represents a single participant (user or organisation), and includes their information, and services within a distributed collection of CSS Nodes. It provides both a pervasive capability and a social networking capability in an integrated form. A CSS can be associated to zero or more Community Interaction Spaces (CIS), which are a representation of multi-participant community. A CSS can interact, communicate, or share directly with another CSS, not mediated by a CIS.

CROWD COMPUTING
Crowd computing is an overarching term that defines the plethora of human interaction tools that enable idea sharing, non-hierarchical decision making and the full utilisation of the world's mind space. Examples of these tools (many falling under the Web2.0 umbrella) include collaboration packages, information sharing software, such as wikis, blogs, alerting systems, social networks, SMS, MMS, Twitter, Flickr, MS SharePoint, and even mashups.

Reference:

DATA PROTECTION LEGISLATION
For an annotated list of the data European Union protection laws and directives, as well as pdf files of their contents, see the website of EDPS: European Data Protection Supervisor:

DATABASE MANAGEMENT SYSTEM
A database management system (DBMS) is a set of software tools designed and implemented in order to assist the maintenance and utilisation of large collections of data.

Reference:

DEPENDABILITY
Dependability is a value showing the reliability of a person to others because of his/her integrity, truthfulness, and trustfulness, traits that can encourage someone to depend on him/her. Dependability as applied to a computer system is defined as the trustworthiness of a computing system which allows reliance to be justifiably placed on the service it delivers.

Reference:

DEPLOYMENT ARCHITECTURE
A view of the proposed hardware that will make up the new system, together with the physical components that will execute on that hardware. Includes specifications for machine, operating system, network links, backup units &etc.

DEPLOYMENT MODEL
A model of the system as it will be physically deployed

DEVICE MANAGEMENT
Management of any sensor, actuator or display device that can be connected to a CSS node and manipulated or queried by the device manager drivers. Examples of such devices are RFID Sensors, display monitors, pressure pad sensors, actuators etc.(see Sensor, and Actuator).
**Digital Identity Management**

Identity management means managing various partial digital identities (usually denoted by digital pseudonyms) of an individual person, i.e., administration of identity attributes including the development and choice of the partial digital identity and digital pseudonym to be (re-)used in a specific context or role. Identity management is called privacy-enhancing if it sufficiently preserves unlinkability (as seen by an attacker) between the partial digital identities of an individual person. It involves the process of digital identity selection and the processes related with digital identity providers.

Reference:

---

**Digital Identity Provider**

A digital identity provider is an entity which provides basic services regarding the support of partial digital identities, namely authentication and identifier/pseudonym management, within some identity domain. In "Laws of Identity", Kim Cameron chooses to not use the very generic "attribute values" used by Pfitzmann and Hansen, and instead defines digital identity as "a set of claims made by one digital subject about itself or another digital subject". In this terminology, a digital identity provider is a digital subject that provides authentication and identification claims relatively to another digital subject. Relying parties that trust a digital identity provider can use such claims to identify the user.

Reference:

---

**Digital Identity Selection**

Digital identity selection is the process of selecting the most appropriate partial digital identity for interacting with some service or resource. For this to happen information related to the various partial digital identities of an individual person has to be available. Consequently, in order to preserve as much as possible the linkability information between different partial digital identities, this process should take place in an environment accessible (as much as possible) only to the user. The most appropriate partial digital identity is typically the one that minimizes partial digital identity linking risk and data disclosure while satisfying the requirements of the operation to be performed on that service or resource.

Reference:

---

**Direct Trust**

A trust relationship derived from the experiences of direct interactions between two parties can be characterised as direct trust.

Reference:
**Disaster Management**

Disaster management is a process or strategy that is implemented when any type of catastrophic event takes place. Sometimes referred to as disaster recovery management, the process may be initiated when anything threatens to disrupt normal operations or puts the lives of human beings at risk. Governments on all levels, as well as, many businesses create some sort of disaster plan that make it possible to overcome the catastrophe and return to normal function as quickly as possible.

Reference:

**External Service**

A service that has been developed independently from SOCIETIES and has no technical or functional dependency with it.

**Federated Identity**

In a digital identity provider scenario we have trust relationships between the relying parties and the digital identity providers. In a federated identity scenario we also have trust relationships between different digital identity providers. This enables entities using partial digital identities from different identity domains to interact.

**GUI**

Graphical User Interface

**Group**

Is defined as: “An assemblage of persons or objects gathered or located together; an aggregation: a group of dinner guests; a group of buildings.”

Reference:
http://www.thefreedictionary.com/group

**Hardware Manufacturers**

A business role: These are device manufacturers, who are responsible for new intelligent devices or upgrading existing ones that could support the new applications, and sensor manufacturers, who are responsible for building, supplying or adjusting sensors to support context-aware applications.

**Identifiability**

Identifiability of a subject from an attacker's perspective means that the attacker can sufficiently identify the subject within a set of subjects, the identifiability set. Identifiability is the negation of anonymity.

Associated with this concept is usually the identifier, which is a value that by itself identifies the subject from the set of all the subjects.

Reference:

**Identity**

Pfitzmann and Hansen define identity as any subset of attribute values of an individual person which sufficiently identifies this individual person within any set of persons. So usually there is no such thing as "the identity", but several of them. Identity enables both to be identifiable as well as to link (some) Items Of Interest (IOIs - messages, information, ...). Formally speaking, identity can be explained and defined as a property of an entity in terms of the negation of anonymity and the negation of unlinkability.

Philosophically, Identity is what defines if one entity is the same as other or not. It is also a social construct of how we see ourselves and, probably more importantly, how others see us. This concept of identity distinguishes between "I" and "Me". For more on this refer to George H. Mead: Mind, Self and Society; Chicago Press 1934

Reference:

IDENTITY DOMAIN

An identity domain is a subject identifier namespace administered by one entity, typically the digital identity provider. Subjects can authenticate with the administering entity w.r.t. identifiers that belong to the domain it administers. Subjects that are authenticated with the administering entity inherently trust it to provide claims about other subjects authenticated in the same domain. They may perform the role of relying parties towards the trusted party that authenticates them.

IDENTITY SELECTION PREFERENCE

Identity Selection Preferences are used during the Digital Identity Selection process and suggest whether an identity should be used under the current context for a specific transaction. An identity selection preference is a type of privacy preference. An Identity Selection Preference is related to only one identity.

Reference:
http://www.iiiaijournals.org/security/sec_v2_n1_2009_paged.pdf
http://doi.ieeecomputersociety.org/10.1109/ICONS.2008.46

INDIRECT TRUST

A trust relationship or a potential trust relationship built from recommendations by a trusted third party or a chain of trusted parties (trust path) is called indirect trust.

Reference:

INFRASTRUCTURE PROVIDERS

A business role: These are stakeholders that either provide the SOCIETIES platform or infrastructure that integrates with the SOCIETIES platform and/or provides data to the SOCIETIES platform. Examples include: CIS/SOCIETIES platform providers, CSS interactive devices providers, CSS providers, Conference CIS providers, Connectivity providers, Network infrastructure providers, etc.

INTEGRATOR

An integrator tests and validates the platform and third party services by specifying, implementing and executing integration tests. It is a technical role preformed by a person or group of people.

INTEROPERABILITY

Interoperability is a property referring to the ability of diverse systems and organizations to work together (inter-operate).

Reference:

LEARNING

Learning is acquiring new knowledge, behaviours, skills, values, preferences or understanding, and may involve synthesising different types of information. In computer science, it refers to the computer systems' ability to evolve behaviours based on empirical data, such as from sensor data or databases. A system self improves by employing mechanisms that allow it to learn how to tailor its behaviour to better meet the needs of the individual user based on previous user behaviour and contextual states.
LOCATION (PHYSICAL)
A specific instance of dynamically derived context information that defines the physical whereabouts of the corresponding entity.

MANUFACTURER
The team that produces, develops and maintains the SOCIETIES platform.

MOBILE OS PROVIDERS
A business role: These companies are responsible for producing mobile operating systems that are capable of supporting or even make easier for the user to take advantage of the SOCIETIES features.

MULTI-DOMAIN FEDERATED IDENTITY
In a single identity domain scenario we have trust relationships between the digital subjects that authenticate within some domain and the entity that administers that domain, typically a digital identity provider. This is typically referred to as a form of federation in identity management. However, in a scenario with multiple identity domains openly communicating, the implications of the concept of federation become slightly different. Trust relationships between the different domain administering entities become necessary, first and foremost to transmit claims that identify subjects across these identity domains.

NODE
A node is a physical device or virtual machine instance, which is capable of executing deployed services.

NODE TYPE
A node-type is a logic grouping of deployable services/components that provide a well defined set of functionality, when deployed to a Node. More formally, a Node type is a set of software components with related technologies that provide a set of well defined functionality given via interfaces and usage patterns.

ONLINE COMMUNITY
An online community is a virtual community that exists online whose members enable its existence through taking part in membership rituals. An online community can take the form of an information system where anyone can post content, such as a Bulletin board system or one where only a restricted number of people can initiate posts, such as Weblogs. Online communities have also become a supplemental form of communication between people who know each other primarily in real life.

Reference:

OPERATING SYSTEM
Is software that runs on computers, it manages its hardware resources, and it provides common services for the execution of application software.

ORGANISATIONS
A Business role: These are existing stakeholders that are expected to gain profit by introducing CIS into their business. These can be, for example, small-medium businesses that want to better identify relevant customers and provide additional focused services and benefits to them or aid organizations that can gain from faster assessment of situations using wisdom of the crowd. Example stakeholders: public transport companies, sport facility centres, theatre-halls, conference centres, government agencies, centres for satellite based crisis information, end user’s companies, restaurants, university management offices, sponsor companies, etc.
**PARTIAL DIGITAL IDENTITY**

A partial identity is a (linkable) subset of attribute values of a complete identity, where a complete identity is the union of all attribute values of all identities of this person. Whereas we assume that an "identity" sufficiently identifies an individual person (without limitation to particular identifiability sets), a partial identity may not do so. A partial digital identity is the digital representation of a partial identity. Whenever we speak of digital identity, we peak of partial digital identity since the strict definitions of digital identity and complete digital identity are not useful.

A digital pseudonym might be an identifier for a partial digital identity. Re-use of the partial identity with its identifier(s), e.g., a pseudonym, supports continuity in the specific context or role by enabling linkability with, e.g., former or future messages or actions. Furthermore, it provides the possibility to authenticate w.r.t. the partial digital identity.

**Reference:**

**PARTICIPANT**

It is used to refer to a single user or organisation which is contained within a collection.

**PERSONAL DATA**

Personal Data According to Article 2 (a) of Regulation (EC) No 45/2001: "Any information relating to an identified or identifiable natural person, referred to as "data subject" - an identifiable person is someone who can be identified, directly or indirectly, in particular by reference to an identification number or to one or more factors specific to his or her physical, physiological, mental, economic, cultural or social identity".

The name and the social security number are two examples of personal data which relate directly to a person. But the definition also extends further and also encompasses for instance e-mail addresses and the office phone number of an employee. Other examples of personal data can be found in information on physical disabilities, in medical records and in an employee's evaluation. Personal data which is processed in relation to the work of the data subject remain personal/individual in the sense that they continue to be protected by the relevant data protection legislation, which strives to protect the privacy and integrity of natural persons. As a consequence, data protection legislation does not address the situation of legal persons (apart from the exceptional cases where information on a legal person also relates to a physical person).

**Reference:**
EDPS: European Data Protection Supervisor, Data Protection Glossary on-line: http://www.edps.europa.eu/EDPSWEB/edps/site/mySite/pid/84#personal_data

**PERSONAL SMART SPACE**

A Personal Smart Space (PSS) is defined by a set of services within a dynamic space of connectable devices where the set of services are owned/controlled or administered by a single user or organisation. It facilitates interactions with other PSSs, is self-improving and capable of pro-active behaviour. A PSS has the following characteristics:

- it is user centric and can move with the user.
- it has an "owner", the person or legal entity on whose behalf the smart space operates.
- it respects the privacy of the user and guards personal information disclosure.
- it allows services and applications to be shared with other PSSs.
- it supports operation in an ad-hoc environment.
- it learns from previous interactions, to partially automate recurring tasks or actions or suggest alternates.
- it allows 3rd party applications deployed to a PSS to adapt to the current situation and context.

**Reference:**
Personalisation involves the use of technology to accommodate the differences between individuals and the unique characterization of a person's needs and expectations from an environment/situation. Personalisation is the set of processes that adapt the behaviour of a system so it appears differently to different users or to the same user in different contexts. By "appears" we mean more than just the colour of the screen but the way in which the system reacts to the user. This includes the services it selects, chosen devices, how services are manipulated at runtime and any automatic triggering it may do on the user's behalf.

Reference:

Pervasive Infrastructure

Pervasive infrastructure is the hardware and software requirements in order to fulfil the pervasive computing paradigm centred on a high-level conceptual model consisting of devices, users, software components and user interfaces. It is a collection of hardware elements and software components provided by possibly different providers, enabling network connectivity, context-awareness, and various businesses or other kind of logic on a relatively large scale, usually hosted by immovable in public places or private corporations.

Reference:
Karen Henricksen, Jadwiga Indulska and Andry Rakotonirainy, "Infrastructure for Pervasive Computing: Challenges"

Pervasive Community

A pervasive community is a group of, two or more, individuals who have agreed to share some, but not necessarily all, of their pervasive resources – personal information, context data, services, devices – with other members of that community. A pervasive community, once constituted, forms a Community Interaction Space (CIS). There is a one-to-one mapping between pervasive communities and CISs. Members of a pervasive community interact with a CIS via their own personal Co-operating Smart Space (CSS). There is a one-to-one mapping between individuals and CSSs. The only way in which an individual can participate in a CIS is via their CSS but they can also interact with other CSSs without having to form pervasive communities or create CISs. Individuals may belong to any number of pervasive communities, and thus CISs, simultaneously. Individuals may also, of course, interact with other individuals without using CSSs by employing more traditional mechanisms. The individual members of a pervasive community do not need to be human beings. They can also be organisations, smart space infrastructures, autonomous or semi-autonomous agents, etc. The key defining characteristic of a pervasive community member is the ability to provide and/or take advantage of pervasive technology. Thus, business enterprises, smart shopping malls, robotic companions, etc. can form pervasive communities with human beings or, indeed, with each other. Pervasive communities can be dynamic in nature, with CISs being formed in an ad hoc fashion as and when required. They can also be created, or become, more permanent and continue to exist even when all the participating members, or their CSSs, are offline. Pervasive communities can also spawn sub-communities or merge with other communities. The pervasive resources which can be shared via a CIS include, service provision, actuator control and both individual and community context information, preferences and behaviours.

Platform

A platform is a set of software components or hardware subsystems, with related technologies that provides a set of precise functionality defined via interfaces and usage patterns. Components supported by that platform can interact and execute without being aware for the details of how the functionality provided by the platform is implemented.
**PLATFORM / CORE SERVICE**

Core Services are functional capabilities exposed by the platform itself. These are services packaged with the SOCIETIES platform, by the platform manufacturer.

**POINT OF INTEROPERABILITY**

A Point of Interoperability (POI) is defined as a point in the architecture where information between two interoperable components is exchanged.

**PREFERENCE**

Preference could be conceived of as an individual's attitude towards a set of objects, typically reflected in an explicit decision-making process. Alternatively, one could interpret the term "preference" to mean evaluative judgment in the sense of liking or disliking an object which is the most typical definition employed in psychology. However, it does not mean that a preference is necessarily stable over time. Preference can be notably modified by decision-making processes, such as choices, even in an unconscious way.

Reference:

**PRIVACY**

Privacy is the ability of an individual or group to seclude themselves or information about themselves and thereby reveal themselves selectively. In computer science, privacy concerns exist wherever uniquely identifiable data relating to a person or persons are collected and stored. In some cases, these concerns refer to how data is collected, stored, and associated. In other cases the issue is who is given access to information. Other issues include whether an individual has any ownership rights to data about them, and/or the right to view, verify, and challenge that information.

Reference:

**PRIVACY POLICY**

A privacy policy is a legal document that discloses some or all of the ways a party gathers, uses, discloses and manages a customer's data. The exact contents of a privacy policy will depend upon the applicable law and may need to address the requirements of multiple countries or jurisdictions.

Reference:
http://en.wikipedia.org/wiki/Privacy_policy

**PRIVACY POLICY AGREEMENT**

Privacy Policy Agreement is the document generated after a successful privacy policy negotiation and defines:

- the list of data that the user will disclose to the service,
- what operations (READ, WRITE, CREATE, DELETE) are allowed to be performed on each data item,
- the conditions placed on the service with regard to the processing of each data item such as sharing with 3rd parties, data retention period, user's right to opt-out etc

Both parties keep a copy of the Privacy Policy Agreement for auditing purposes

**PRIVACY POLICY NEGOTIATION**

Privacy policy negotiation is the process of making a mutual agreement between the privacy policy of a service provider and the privacy policy of a service consumer.

Reference:
http://www.w3.org/2006/07/privacy-ws/papers/22-hatakeyama-negotiation-attributes/
http://www.w3.org/2006/07/privacy-ws/papers/24-preibusch-negotiation-p3p/
**PRIVACY PREFERENCE**

A Privacy Preference expresses the user's wishes with regards to his privacy, that is the disclosure of his personal data and their processing after disclosure. A Privacy Preference can be dependent on the user's context and the requestor's trust evaluation.

**PROACTIVITY**

Pro-activity is the ability to being anticipatory and taking charge of situations, i.e. acting in anticipation of future problems, needs, or changes. Pro-activity should take decisions on behalf of the user in order to personalise services and environments.

**Reference:**

**QUALITY OF CONTEXT**

Quality of Context (QoC) denotes a set of parameters, used to describe the quality of certain Context Information elements. Examples of QoC parameters include: Accuracy, Probability of Correctness, Reliability, Resolution, Timeliness, Frequency, Price, etc.

**Reference:**

**REASONING**

Reasoning is the process of looking conclusions or actions. In computer science, reasoning is the area dedicated to understanding different aspects of reasoning in a way that allows the creation of software which allows computers to reason completely or nearly completely, automatically.

**Reference:**

**RELIEF WORKER**

These end-users are assessment experts who arrive at a disaster scene, in order to assess damage, available resources, and so help coordinate the disaster relief actions. They are supported by experts located off-site (i.e. away from the disaster area) that coordinate the assessment from operation centres.

**REPUTATION**

Reputation is the opinion (more technically, a social evaluation) of the group of entities toward a person, a group of people, or an organisation on a certain criterion. It is an important factor in many fields, such as education, business, online communities or social status. Reputation can be considered as a component of the identity as defined by others.

**Reference:**

**RESOURCE**

Is a service with a maximum constraint on the number of concurrent users. For example, a display may only be used by one user at a time. Thus it is termed a resource, but still is represented by a service. (Other physical constraints of a resource are handled by the service properties, and runtime behaviour) See Resource sharing for a more information.

**RESOURCE SHARING**

Resource sharing is required when two systems attempt to use the same service in parallel. It may be the case that only one system can use the service at a time in which case access to the resource must be shared. It can also be the case that multiple systems can use the service at a time but personalisation must be mediated to adapt the service to best meet the needs of all systems involved.

Resource sharing systems include peer to peer computing, utility computing, cluster computing, autonomic computing and grid computing, as well as, their field of applications. In pervasive computing, resource sharing implies the use of common computational or device sources such as processing power, storage, network bandwidth, printer usage, etc.
Within SOCIETIES, a CSS can be thought of as a system, and the term used to represent the following behaviour: to grant specific usage rights to the specified service from other CSSs; and to announce the availability of a specified service to other CSSs.

Reference:

SECURITY
Computer security is a branch of computer technology known as information security as applied to computers and networks. The objective of computer security includes protection of information and property from theft, corruption, or natural disaster, while allowing the information and property to remain accessible and productive to its intended users. The term computer system security means the collective processes and mechanisms by which sensitive and valuable information and services are protected from publication, tampering or collapse by unauthorised activities or untrustworthy individuals and unplanned events respectively. The strategies and methodologies of computer security often differ from most other computer technologies because of its somewhat elusive objective of preventing unwanted computer behaviour instead of enabling wanted computer behaviour.

Reference:

SENSOR
A sensor is a device that measures a physical quantity and converts it into a signal which can be read by an observer or by an instrument.

Reference:

SENSOR NETWORK
A sensor network consists of spatially distributed autonomous sensors to cooperatively monitor physical or environmental conditions, such as temperature, sound, vibration, pressure, motion or pollutants, etc.

Reference:

SERVICE
A service is a mechanism to enable access to one or more capabilities, where the access is provided using a prescribed interface and is exercised consistent with constraints and policies as specified by the service description.

SERVICE CONSUMERS
A business role: The end users or entities using the final products and services.

SERVICE MANIFEST
The Service Manifest is a machine readable information for 3rd party services and persists everything that is relevant to fully describe a service to the SOCIETIES platform.

SERVICE MARKETPLACE
A searchable index for CSSs when looking for services contained in a well-known Service Store

SERVICE PRODUCER
A business role: The developer or team of developers responsible for the production of a service. (see External Service, Third party Service).
SERVICE PROVIDER
A business role: These are stakeholders who make available software services for use by other parties. The can also act as service producers. These can be high-level services, addressed directly to end users, or low-level, for example middle-ware that enables the creation, deployment, and provision of complicated services. Example stakeholders: Enterprises or associations that want to provide services to affiliates. (see External Service, and Third Party service).

SERVICE-ORIENTED ARCHITECTURE
A service-oriented architecture (SOA) is essentially a collection of services. These services communicate with each other. The communication can involve either simple data passing or it could involve two or more services coordinating some activity. Some means of connecting services to each other is needed.
Reference: http://www.service-architecture.com/web-services/articles/service-oriented_architecture_soa_definition.html

SMART HOME INTEGRATORS
A business role: The companies building or equipping smart homes and deploying smart home services, providing their expertise to unify interoperability issues between devices and sensors from different hardware providers, services from different service providers, and a variety of technologies, offering ease of use and joined up solutions to the end user.

SMART SPACE
A smart space is a collection of devices and software components which enable context-awareness and pro-actively adapt to user's preferences.

SOCIAL NETWORK PROVIDERS
A business role: Companies or individuals who operate an existing social network site e.g. Facebook, or new emerging social network sites, e.g. Diaspora. (see also Social Network Site)

SOCIAL NETWORK SITE
A social networking site (or service) is an electronic "place" where people interact and implement the principles of social computing, by employing and/or using services and participate in events and diverse activities.

SOCIAL COMPUTING
Social computing is a general term for an area of computer science that is concerned with the intersection of social behaviour and computational systems. It is used in the following two ways. In the weaker sense of the term, social computing has to do with supporting any sort of social behaviour in or through computational systems. It is based on creating or recreating social conventions and social contexts through the use of software and technology. Thus, blogs, email, instant messaging, social network services, wikis, social bookmarking and other instances of what is often called social software illustrate ideas from social computing, but also other kinds of software applications where people interact socially. In the stronger sense of the term, social computing has to do with supporting "computations" that are carried out by groups of people. Examples of social computing in this sense include collaborative filtering, online auctions, prediction markets, reputation systems, computational social choice, tagging, and verification games.
SOFTWARE PROVIDER
A business role: These are stakeholders who make available software services for use by other parties. The can also act as service producers. Example stakeholders: Enterprises or associations that want to provide services to affiliates. (see External Service, and Third Party service)

SYNCML
Synchronization Mark-up Language

TELCO OPERATORS
A business role: They provide the network infrastructure for supporting middleware and services.

THIRD PARTY SERVICE
An extension or utilisation mechanism that can be used to add applications which interact with, or extend the services of the SOCIETIES platform. These are developed independently from the SOCIETIES platform and may have different licensing conditions. However, they have a technical or functional dependency on the SOCIETIES platform.

THIRD PARTY SERVICE PROVIDER
A business role: In the SOCIETIES context these are companies or individuals providing sets of 3rd party services that are integrated with the SOCIETIES platform. Such companies are usually operate in a specific business domain. Examples of these services include, electronic medical records services, emergency services, social shopping services, community service providers, etc.

TRUST
Trust is the firm belief in the competence of an entity to act dependably, securely, and reliably expectation an agent has about another’s future behavior based on the history of their encounters.

Reference:

TRUST MANAGEMENT
Trust management was defined by Grandison as the "activity of collecting, encoding, analyzing and presenting evidence relating to competence, honesty, security or dependability with the purpose of making assessments and decisions regarding trust relationships".

Reference:

UBQUITOUS COMPUTING
Ubiquitous computing (ubicomp) is a post-desktop model of human-computer interaction in which information processing has been thoroughly integrated into everyday objects and activities. In the course of ordinary activities, someone "using" ubiquitous computing engages many computational devices and systems simultaneously, and may not necessarily even be aware that they are doing so. This model is usually considered an advancement from the desktop paradigm.
This paradigm is also described as pervasive computing, ambient intelligence, where each term emphasises slightly different aspects. When primarily concerning the objects involved, it is also physical computing, the Internet of Things, haptic computing, and things that think. Rather than propose a single definition for ubiquitous computing and for these related terms, a taxonomy of properties for ubiquitous computing has been proposed, from which different kinds or flavours of ubiquitous systems and applications can be described.

Reference:
Use Case

A Use Case represents a discrete unit of interaction between a user (human or machine) and the system. A Use Case is a single unit of meaningful work; for example, creating a train, modifying a train, and creating orders are all Use Cases. Each Use Case has a description which describes the functionality that will be built in the proposed system. A Use Case may 'include' another Use Case's functionality or 'extend' another Use Case with its own behavior. Use Cases are typically related to 'actors'. An actor is a human or machine entity that interacts with the system to perform meaningful work.

User

Further qualified as:

- End-user who has CSS related technology installed on their devices and use SOCIETIES and its services to interact with others via CIS's or CSS's directly. An end user can be specialised further for a particular domain (e.g., see Relief worker).
- Developer - See service producer definition, can be specialised further depending on the type of service (e.g., see Third Party Service, see External Service, or see Manufacturer for services included in the platform).
- Organisation - A legal entity or group of people with a common identity and defined purpose.
- Company - A legal entity with a common identity and affiliation (e.g., group of employees).
- Integrator - See Integrator.

User Intent

The desired purpose, goal or aim of an end user's behavior (set of actions). Combining the user's past actions with context snapshots can permit the discovery of past, and prediction of future end user goals. Within SOCIETIES this is an estimation of the end user's true intent, which may never be divulged to the system.