Abstract

The SOCIETIES project supports the creation of Ambient Intelligence Communities by discovering, connecting and organising relevant people and things from both physical and digital environments. SOCIETIES uses pervasive technologies to form adaptive communities, while leveraging social networks and crowd computing techniques. Work package 5 covers five distinct areas in which we have researched, designed and prototyped supporting technologies for the SOCIETIES CSS Individual and Community Experience. They are: Intelligent Community Orchestration, Context, Personalisation, Privacy & Trust and User Agent. This document provides a summary on the work of the SOCIETIES CSS Individual and Community Experience work package and a short analysis of the final user trial evaluations including recommendations for further research and work.
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] WP5 CSS Individual and Community Experience
[Editor: Name, company] Eliza Papadopoulou, HWU
[Work-package leader: Name, company] Nick Taylor, HWU
[Estimation of PM spent on the Deliverable] 4 PMs

Copyright notice
© 2014 Participants in project SOCIETIES
# List of authors

<table>
<thead>
<tr>
<th>Company</th>
<th>Author</th>
</tr>
</thead>
<tbody>
<tr>
<td>HWU</td>
<td>Eliza Papadopoulou, Stuart Whyte, Diana Bental, Nick Taylor</td>
</tr>
<tr>
<td>TSSG</td>
<td>John McGovern</td>
</tr>
<tr>
<td>ICCS</td>
<td>Nicolas Liampotis, Ioanna Roussaki, Nikos Kalatzis</td>
</tr>
<tr>
<td>TRIALOG</td>
<td>Bruno Jean-Bart, Olivier Maridat</td>
</tr>
<tr>
<td>ITSUD</td>
<td>Daqing Zhang, Haoyi Xiong, M’hamed Abdallah</td>
</tr>
<tr>
<td>SETCCE</td>
<td>Mitja Vardjan</td>
</tr>
<tr>
<td>SINTEF</td>
<td>Bjorn Magnus Mathisen</td>
</tr>
<tr>
<td>TI</td>
<td>Luca Lamorte</td>
</tr>
</tbody>
</table>

# List of Code Contributors

<table>
<thead>
<tr>
<th>Company</th>
</tr>
</thead>
<tbody>
<tr>
<td>HWU</td>
</tr>
<tr>
<td>TSSG</td>
</tr>
<tr>
<td>SINTEF</td>
</tr>
<tr>
<td>ICCS</td>
</tr>
<tr>
<td>AMITEC</td>
</tr>
<tr>
<td>DLR</td>
</tr>
<tr>
<td>IBM</td>
</tr>
<tr>
<td>TRIALOG</td>
</tr>
<tr>
<td>ITSUD</td>
</tr>
<tr>
<td>PORTUGAL TELECOM</td>
</tr>
<tr>
<td>SETCCE</td>
</tr>
<tr>
<td>NEC</td>
</tr>
<tr>
<td>PTIN</td>
</tr>
</tbody>
</table>
# Table of Contents

1. Introduction ....................................................................................................................................................................................... 7

2. Prototype Features .................................................................................................................................................................................. 8
   2.1 Intelligent Community Orchestration (John) .......................................................................................................................... 8
      2.1.1 Community Recommendation Manager .......................................................................................................................... 9
      2.1.2 Model Analyser ................................................................................................................................................................. 11
      2.1.3 Data Collectors ................................................................................................................................................................. 13
      2.1.4 Collaboration Pattern Analyser .................................................................................................................................. 14
      2.1.5 Suggested Community Analyser .................................................................................................................................. 15
      2.1.6 Egocentric Community Analyser .................................................................................................................................. 16
   2.2 Context Management ....................................................................................................................................................................... 17
      2.2.1 User and Community Context DB Management .............................................................................................................. 21
      2.2.2 Context Broker .................................................................................................................................................................. 22
      2.2.3 Context Source Management ........................................................................................................................................ 22
      2.2.4 Location Adapter ................................................................................................................................................................. 22
      2.2.5 User and Community Context History Management ................................................................................................... 22
      2.2.6 Context Event Management ........................................................................................................................................ 22
      2.2.7 CIS and CSS Monitoring ................................................................................................................................................... 23
      2.2.8 User and Community Context Inference Management ................................................................................................ 23
      2.2.9 User Context Prediction .................................................................................................................................................. 23
      2.2.10 Community Context Prediction .................................................................................................................................. 23
      2.2.11 User and Community Context Refinement .................................................................................................................. 23
      2.2.12 User and Community Context Inheritance ................................................................................................................. 24
      2.2.13 User Context Similarity Evaluation .................................................................................................................................. 24
      2.2.14 Community Context Estimation .................................................................................................................................. 24
      2.2.15 Context GUI (Luca) ......................................................................................................................................................... 25
   2.3 Personalisation .................................................................................................................................................................................. 26
      2.3.1 Personalisation Manager .................................................................................................................................................. 28
      2.3.2 DIANNE ............................................................................................................................................................................. 28
      2.3.3 User Preference Management ........................................................................................................................................ 29
      2.3.4 Context Aware User Intent - CAUI ................................................................................................................................ 30
      2.3.5 Conditional Random fields based user intent ................................................................................................................ 34
      2.3.6 Social Profiler .................................................................................................................................................................. 37
   2.4 Privacy & Trust .................................................................................................................................................................................... 39
      2.4.1 Privacy Preference Management ...................................................................................................................................... 42
      2.4.2 Privacy Policy Negotiation Manager .................................................................................................................................. 43
      2.4.3 Privacy Data Manager ....................................................................................................................................................... 44
      2.4.4 Privacy Policy Manager .................................................................................................................................................. 44
      2.4.5 Identity Selection Manager .................................................................................................................................................. 46
      2.4.6 Privacy Assessment Manager ........................................................................................................................................ 47
      2.4.7 Privacy Assessment GUI .................................................................................................................................................. 47
      2.4.8 Direct, Indirect and User-perceived Trust Engine ......................................................................................................... 50
      2.4.9 Trust Repository ............................................................................................................................................................... 51
      2.4.10 Trust Evidence Repository and Collector ..................................................................................................................... 51
      2.4.11 Trust Broker and Trust Event Manager ......................................................................................................................... 51
      2.4.12 Trust GUI ........................................................................................................................................................................ 51
   2.5 User Agent ....................................................................................................................................................................................... 53
      2.5.1 Decision Maker ................................................................................................................................................................. 53
      2.5.2 Conflict Resolution ............................................................................................................................................................ 53
      2.5.3 User Feedback ................................................................................................................................................................. 54
      2.5.4 User Action Monitor ...................................................................................................................................................... 57
   3. Prototype Evaluation ........................................................................................................................................................................ 58
      3.1 Lessons Learnt .......................................................................................................................................................................... 58
1 Introduction

The SOCIETIES project supports the creation of Ambient Intelligence Communities by discovering, connecting and organising relevant people and things from both physical and digital environments. SOCIETIES uses pervasive technologies to form adaptive communities, while leveraging social networks and crowd computing techniques. Work package 5 covers five distinct areas in which we have researched, designed and prototyped supporting technologies for the SOCIETIES CSS Individual and Community Experience. They are: Intelligent Community Orchestration, Context, Personalisation, Privacy & Trust and User Agent.

Intelligent Community Orchestration provides the orchestration mechanism for dynamic community creation and membership. It facilitates the operation of a variety of intelligent systems (context, personalisation, trust) to work across all of the CSSs in a given community.

Context provides functionality regarding the representation, collection, management and sharing of context information and the inference of higher level context attributes, in order to support the provision of context-aware experience to the CSS users. It extends that functionality by leveraging the context of other members of the communities to which a user belongs.

Personalisation supports the complete dynamic management of user preferences and behaviour intent models. This includes monitoring for updates due to context and user behaviour changes, evaluation of preferences and intent models and the application of the evaluated outcomes. It also provides a base functionality which is capable of acquiring user preferences and intent models both explicitly (manually) and implicitly (learning). A range of different learning techniques have been employed. These include benchmark data mining algorithms as well as innovative incremental techniques. To enable the user to manually create and maintain behaviour information, all behaviour models and preferences are translatable into human understandable form allowing the user to view and manipulate the data. The community concept is leveraged as well in order to aid the individual user based on community behaviours.

Two types of privacy protection have been investigated; a priori privacy protection, before data are disclosed to another CSS within a community and a posteriori privacy protection, after the data has been disclosed. Based on these privacy mechanisms, the task has designed, implemented and evaluated components to provide dynamic personalised privacy using privacy preferences, privacy preference learning, data obfuscation techniques, personalized identity selection and privacy policy negotiation. Privacy enforcement has been investigated to provide auditing mechanisms for the evaluation of actual privacy practices. A trust model has also been developed to support collaborations between various stakeholders and roles of CSSs as well as the necessary infrastructure for maintenance and management of dynamically changing trustworthiness of stakeholders with respect to different domains of collaboration.

The user agent coordinates and manages the implementation of behaviours on behalf of the user. Therefore it provides various decision making functionalities with the aim of mitigating the need for user intervention. Conflicts are dealt with in terms of discrepancies between internal intelligent systems as well as dissimilarities in the needs of the user as a community member and as an individual. The user is provided with the opportunity to influence all decision making procedures through action monitoring and feedback mechanisms.

This document provides a summary on the work of the SOCIETIES CSS Individual and Community Experience work package and a short analysis of the final user trial evaluations including recommendations for further research and work.
2 Prototype Features

2.1 Intelligent Community Orchestration (John)

The goal of the Intelligent Community Orchestration (ICO) is to aid an end-user in the orchestration of the SOCIETIES platform allowing them to get the most relevant community (or communities) for them, as and when they need it. To facilitate this goal a multifaceted approach has been adopted in Task 5.1 to provide the best possible outcome for the end-user.

Figure 1 shows how the granularity of the data under inspection has an effect on the potential precision of the results, while also impacting on the privacy level of the end-users data. Approach 1 takes a holistic view of the available system data, requiring low privacy settings to enable sufficient data gathering for decision making. Whereby, Approach 3 is working on behalf of a single end-user within the boundaries of that end-user’s data. The approaches can be described as follows;

1) **Approach 1**: provided by the CSM Analyser converts raw data into context state models (CSM). Each CSM is then placed into the CSM plane where trajectory data mining techniques will be used to identify the effect the CSM creates within the model and recommend various actions based on the analysis.

2) **Approach 2**: provided by the Collaboration Pattern Analyser, which is responsible for tracking the behavior of the end-users in the community, and how they use the tools offered to them for collaboration and coordination, in order to suggest a change in the community characteristics or use of these tools.

3) **Approach 3**: provided by the Egocentric Community Analyser, which operates from within an individual CSS, and analyses data accessible by that CSS (such as personal data and revealed data from other CSSs) in order to identify potential new communities and sub-communities that could be automatically created or suggested to that CSS’s end-user, as well as to identify configurations for existing communities and to delete existing communities.

4) **Approach 4**: provided by the Pervasive User Interface (PUI) will develop a toolkit for constructing a physical user interaction layer for social and collaborative applications, with a special focus on manually creating new CISs. The toolkit will provide a set of primitives that will allow the end-user to select CSSs (e.g. by bumping phones), to select CISs (e.g. QR codes, augmented reality) and creating and joining CISs.

![Figure 1 Intelligent Community Orchestration Mining Overview](image-url)
A rudimentary walkthrough of the architecture of the ICO components can be split into both reactive and interactive.

- Reactive intelligent components are listening for events triggered within the system. Once these events are recognised they are captured by one of the data collector components (ref. 2.2.3 Data Collectors) which then passes these to the respective intelligent component(s) which analysis the event and its consequences on the system.

- Interactive intelligent components respond to an end-user initiation either on demand or via a timed batch process to perform an assessment of an end-user’s current community status (ref. 2.2.1 Community Recommendation Manager and 2.2.6 Egocentric Community Analyser).

### 2.1.1 Community Recommendation Manager

The Community Recommendation manager is responsible for providing a list of CIS suggestions based on a filter provided by the user or a 3rd party service. The component uses the CIS Management to retrieve the remote communities which the CSS user still not participates presenting the results according to the selected filters.
The CRM enables the CSS users or a 3rd party service, through an API, to search for communities by defining a set of primary or secondary filter. The primary filter allows querying accurate results, thus only results that exactly match the filter will be presented. The secondary filter allows queries which will present precise or close results.

The communities have by default a number of membership criteria associated which expose the purpose of their creation. The membership criteria present their information using context attributes available in the Context Management component. The attributes can be numeric or string values that are used a posteriori for comparisons in the CRM.

Each filter uses a set of operators provided by the CRM which are used for comparisons. The filters compare the context attributes in the membership criteria with the data provided by the user or service. The filter operators possible are: equal, not equal, greater, greater or equal, less, less or equal and in. For instance, a filter can be defined as: location, equal, Tramore. While a second filter can defined as: age, greater or equal, 25. The results would present communities located in Tramore with people above or equal to 25 years.

In addition, the CRM results are shown ranked from most to least relevant, based on the filters defined for search. In addition, there is a limit by default to five results. The limit can be changed at any time by requestor.

The figure below presents a more detailed the internal structure class of the CRM component.
2.1.2 Model Analyser

The premise behind the Model Analyser is to use context updates to provide a real time interaction between an end-user’s pervasive environment and their interaction within the platform. The Model Analyser is a collective of dependant components the function of which is converting raw data into context state models.
2.1.2.1 Context State Models (CSM) Analyser

CSM Analyser provides modelling of the context data into roles that will be used to clean the data in preparation for the mining algorithm. Event updates received from an end-user trigger the process, events can be of a number of formats such as:

- Adding Context attributes, for instance an end-user adds ‘Experience’ to their professional profile.
- Updating Context attributes, some attributes such as location will be regularly updated.
- Manually selecting a new communities, an end-user wishes to join a specific community from a GUI of selectable communities.
As event updates are received, the impact is analysed and then checked if there is any consequential ripple effects caused by the event. For instance, an updated context event would have the immediate effect of potentially removing an end-user’s acceptable criteria for joining a community. The ripple effect would check if the end-user’s new update allows them to join additional communities, or does this event trigger a tipping point for a new community suggestion.

- Available on Virgo. No need on Android.

### 2.1.2.2 Group Identifier

The Group identifier provides the mechanism to identify how changes to the CSM can aid in the identification of theoretical groups that may form the basis for a community and the corresponding community management action.

- Available on Virgo. No need on Android.

### 2.1.3 Data Collectors

There are two further components in ICO namely the ‘CIS Data Collector’ and the ‘CSS Data Collector’. These are simply there to aid in good software design principles; they are task 5.1 common listeners and publishers of activity data within the SOCIETIES platform. For instance, Collaboration Pattern Analyser and Group Identifier components subscribe to CIS Data Collector rather than both components developing the similar subscribers.

- Available on Virgo. No need on Android.
2.1.4 Collaboration Pattern Analyser

![Diagram of Collaboration Pattern Analyser](image)

The Collaboration Pattern Analyser (CPA), part of the Computer Supported Co-operative Work (CSCW) sub module, is tasked with continuously analysing activities within a CIS to investigate if said activities suggest the creation of a new CIS. In this case the CPA would send the member list of this new CIS as a CIS suggestion to the Suggested Community Analyser. The CPA uses the continuous activities to build and maintain a social graph. This graph is also accessible for other SOCIETIES platform components. More specifically the graph feeds into the other CSCW components of Intelligent Community Orchestration; User Presence Manager and CIS Member Priority Manager.

Additionally the CPA also looks for trends within the CIS activities. These trends are then monitored, thus keeping a log of how the trends are changing. CPA provides a REST interface to get the history of each trend, also to get a graph of N trends. This functionality was tested with the SOCIETIES app distributed during the ICT2013 conference. The figure below shows an example of mined trends from 11 of October 2013, around the time when the Nobel Peace Prize winner was announced. The trends was mined from an activityfeed where the activities was simulated with tweets hashtagged with “#bbc” as raw data.
2.1.5 Suggested Community Analyser

The Suggested Community Analyser (SCA) receives suggestions from sub modules of 5.1, namely the Egocentric Community Analyser (ECA), Collaboration Pattern Analyser (CPA) and the Context State Model (CSM) Analyser. The suggestions that the SCA receive are in the form of a Community Suggestion Model.

Figure 8 Mined trends during the 2013 ICT conference.

Figure 9 Community Suggestion Model
The Community Suggestion Model contains information to allow the SCA to know what action is being suggested and concerning which CIS. It also describes how the component which sent the suggestion came to its conclusion and who else (if anyone) is affected if the suggested action is executed. These actions can be creating a CIS, joining a CIS, leaving a CIS or deleting a CIS.

When the SCA receives a suggestion, it performs basic checks to ensure that the action which is recommended can be performed. For example, in the case that a ‘delete’ suggestion is received, the SCA will ensure that the CIS in question exists, and that the current user that the SCA is running on behalf of is the owner of that community. If the initial checks pass, the SCA will issue a User Feedback notification to allow the user to respond to the suggestion. On a positive response, the SCA will perform the action on behalf of the user. On a negative response, or if the suggestion does not initially pass the checks, the SCA will broadcast an event to alert the sub modules of 5.1 that the user has either rejected the suggestion or that the suggestion cannot be implemented, so they are able to update their algorithms accordingly.

If the action which is suggested affects other users, the SCA which received the suggestion can communicate with the SCA instances running on behalf of the other users to allow them to act accordingly. There are three possible outcomes that follow;

- In the case which the original suggestion recommends to delete a CIS and that user has agreed, the SCA of the affected users will issue a User Feedback notification to alert the user that the CIS no longer exists, and thus they are no longer a member of that CIS.
- In the case which the original suggestion recommends to leave a CIS and that user has agreed, the SCA of the affected users will issue a User Feedback notification to ask the user whether they wish to leave that community as well.
- In the case which the original suggestion recommends to create or join a CIS and that user has agreed, the SCA of the affected users will issue a User Feedback notification to ask the user whether they wish to join that community as well.

### 2.1.6 Egocentric Community Analyser

The Egocentric Community Analyser (ECA) is deployed on each CSS and attempts to identify opportunities for the creation of new communities and manipulation of existing communities that would be of relevance to the host CSS based on the host CSSs view of other CSSs utilising the SOCIETIES platform. The ECA does not use the data collector elements to obtain data for analysis, but rather it is restricted to using only data that its host CSS is able to access. This data includes profile and context data that is made available from friend CSSs and CSSs which are members of the same CISs which the host CSS is a member of.
As such this component attempts to make use of the limited information it has access to, working within the privacy restrictions of the system on behalf of a particular CSS, in order to support community creation, configuration, and deletion. It is egocentric in the sense it is restricted by what the host CSS can access, but it endeavours to identify possibilities that benefit not only the host CSS but as many other CSSs as possible given the restrictions. Also as the component is triggered internally and does not use any direct end-user input, it relies entirely on its own principles for identifying community lifecycle possibilities based on analysis of the data available. The methods used allow the component to make various sorts of recommendations based on a wide variety of factors, while being abstract enough to work without need for end-user intervention.

When possibilities are identified, the ECA sends its suggestions to the Suggested Community Analyser (SCA). The ECA will only receive feedback if its suggestion could not be implemented or the user has rejected the suggestion. In the case where the user has rejected the suggestion, the ECA updates its algorithm to only issue another suggestion if the target CSS’s are more related than the previous suggestion.

2.2 Context Management

The Context Management system (CMS) acts as an intermediate layer between platform/3P context-aware services and the sources of context information. The CMS functionality is realised through a multitude of components available for two main distributions: (i) the Virgo CSS container (Rich Client) and (ii) the Societies Android Application (Light Client). The CMS Rich Client consists of two bundles, namely, Context Management and Location Management. The overall deployment of the CMS architecture is illustrated in the following figure.
The diagrams that follow illustrate the components comprising each distribution along with the required/implemented interfaces. It should be noted that the components of the Context Management bundle have been grouped according to the major concept they manipulate or operate on. More specifically, components that operate on user level (single CSS) are grouped together (see first diagram below), as are those that operate on community level (see second diagram below). Components that operate on either user or community level are depicted in both diagrams. The third diagram below illustrates the overall CSM light client architecture deployed on Android devices. Finally, the fourth diagram below illustrates the high level architecture of the Location Management bundle.
Figure 13 User Context Management Rich Client Architecture
Figure 14 Community Context Management Rich Client Architecture
In the following sections, brief descriptions of the main CMS components are provided.

2.2.1 User and Community Context DB Management

User Context DB Mgmt component is responsible for managing and maintaining context data on each node of the CSS. Additionally, it supports the instantiation and the management of data classes for modelling context information and context metadata such as Quality of Context. The Context Model includes all the classes that model the context information to be retrieved, exchanged, maintained and managed in general in the CSS. This component utilises Hibernate, which is an object-relational mapping (ORM) library, in order to map the Context Model objects into records in a traditional relational database. Finally, the User Context DB Mgmt provides information to the Context Event Management, about any changes that are made. The functionality of Community Context DB Mgmt component is similar to the User Context DB Management but it is deployed only on CIS level. Hence it is responsible for instantiating the context model and managing community context data referring to groups of users.
2.2.2 Context Broker

The Context Broker manages the interaction between the components that gather contextual data and the components or services that request the retrieval of context information from the Context DB. The Context Broker enables the exchange of context data among different CSSs and CISs through the WP4 Communications Framework. In addition, it acts as a gateway to the Context History DB. Context Broker is also providing the ability to register or unregister for context changes notifications from the Context Event Management. Finally, it triggers the intelligent functionality provided by the inference management. A light version of the Context Broker is also available on Android (light user client). It supports a subset of features that are related to the interaction with the Context DB.

2.2.3 Context Source Management

The Context Source Manager component serves as intermediate layer between context sources, and the context database. These sources include external sensors, as well as, context-providing services and devices. The underlying model assumes a certain level of intelligence of context sources or respective driver modules. They have to be able to find the access point to the context source manager and provide information about which information they provide. Concluding, context provisioning requires a context source registration process to allow only supported parties to supply such information and to be able to distinguish different providers for the same context attribute type.

2.2.4 Location Adapter

The Location Adapter component, which is the core of the Location Management component of the CSM, exploits IBM’s Presence Zone Server (PZS) functionality in order to determine the location of the user. The component acts as a wrapper for the PZS and handles the registration process of the CSS nodes to the PZS. Thus, it consists of two sub-components: the wrapper/adapter and the configurator. It acts as a secondary context source and therefore, it exploits the Context Source Manager to forward its location updates to the context database.

2.2.5 User and Community Context History Management

The User and Community Context History Management component provides the necessary mechanisms for persistent storage of historical context data (History of Context - HoC) for individuals and communities. Historic context data are maintained on a database that resides on the cloud node of each CSS or CIS. The component also provides data management functionality in order to store, retrieve and delete historic context data. HoC data are modelled by the HistoricAttribute class which is a reduced version of the ContextAttribute, containing only the necessary data to be used for learning processes. ContextAttributes contain a flag in order to indicate if the data should be stored in the HoC DB or not. The same context history management methods are used for both user and community context. Depending on the context identifier that the context history consumer is utilizing the request is routed to the appropriate user or community history database.

2.2.6 Context Event Management

The context management component provides methods for registering CtxChangeEventListeners in order to listen for context change events. There are two ways to subscriber for context change event notification: based on a context identifier and based on a context type. The subscriber's implementation of the CtxChangeEventListener interface allows reacting to the following event notifications: CREATED,
MODIFIED, REMOVED, UPDATED. The mechanism is common for both user and community context data objects.

2.2.7 CIS and CSS Monitoring

This class provides the necessary mechanisms in order to update user and community context based on CSS and CIS related events. Received events are translated according to context model and persisted in context database. Examples of such events are CSS owner profile changes, new CSS connections, CIS membership changes, etc.

2.2.8 User and Community Context Inference Management

User and Community Context Inference Mgmt component coordinates and controls a set of individual methods for inferring and improving the quality of existing context data. This component provides the intelligence in order for the right inference process to be selected. User and community context inference methods include the prediction of context data, the refinement of existing context data, as well as, the inheritance of context data from the members of the community that the user belongs. In addition, community context estimation techniques are provided for communities of users while mechanisms for context similarity evaluation are provided for individual user context.

2.2.9 User Context Prediction

The User Context Prediction component provides functionality that allows the estimation of current context values but also supports the prediction of future context. More specifically, this component is responsible for performing both long term and short term context predictions for individual user context. The prediction algorithm is based on neural networks statistical methods.

2.2.10 Community Context Prediction

The Community Context Prediction component provides functionality that allows the prediction of future community context, in cooperation with the User Context Prediction component. Community Context Prediction can be achieved directly, using History of Community Context and special algorithms for special cases of community context, or indirectly, feeding User Context Prediction results into Community Context Estimation methods. This component is to be demonstrated in the second trial.

2.2.11 User and Community Context Refinement

The User and Community Context Refinement component’s functionality is transparent for the context consumer and supports both on demand and continuous inference. It builds on a plug-in architecture supporting the dynamic inclusion of refinement rules and refinement Bayesian algorithms. Refinement rules are represented in a formal way, specifying its output and input context attributes and the inference algorithm which is capable of evaluating it, together with algorithm specific information. Based on the general information, the reasoning manager handles the refinement rules and forwards them for evaluation to the dedicated algorithms.
2.2.12 User and Community Context Inheritance

The Context Inheritance component provides functionality for context inheritance in hierarchical communities that will enable passing-on of context information from parent communities to child communities when specific conditions apply. More precisely, the child community will examine the list of parent communities it has originated from, and try to find usable existing pieces of context to fill in the gaps, either directly or after appropriate estimation. Furthermore, the collection of user context data from the community members will be supported and enhanced. For example, a user belonging to a community might inherit some context info from this community and not need to fill up all the info manually.

2.2.13 User Context Similarity Evaluation

An application may need to group users according to similarity in one or more aspects. These aspects are examined by the values of a user’s context attributes. The User Context Similarity Evaluation component provides the functionality to quantify the amount of similarity shared in each of these attributes. Each evaluation consists of two or more users and one or more attributes as required by the referencing application. The context broker is used to obtain the needed context attribute data for each user. The evaluateSimilarity method of User Context Similarity Evaluation takes the following input criteria:

- List of users;
- List of attributes to be used in the evaluation.

This evaluationResult returns three levels of similarity data.

- Overall Estimation – Boolean true/false
  - False indicates there was no similarity bond between this group of users under the context attributes assigned, this will allow a requesting App to waste no further processing on this and move on.
  - True indicates we have found something and data from levels 2 & 3 result will give further insight into what it is.

- Attribute Summary – Each of the attributes that were examined will provide a individual metric of its bond to similarity. Again, a requesting App can tell from examining this level which attributes led to the positive similarity evaluation

- Attribute Breakdown – a more detailed breakdown of the similarity based on the taxonomy of each attribute. For example in a requested list of attributes e.g. “Books & Food” we can tell that a group of users like the genre Horror in their reading tastes and all like pizza food.
  - Available on Virgo. No need on Android.

2.2.14 Community Context Estimation

The Community Context Estimation component allows the extraction of new context data that apply to an entire community of users. The component offers various mechanisms that support the estimation of common context values based on common features, mean value, and majority of preferences. These mechanisms range from simple majority rules, up to complex multidimensional calculations, with methods for the calculation of alphanumerical and geometrical types of community context. The mean value, an example of simple calculations on a set of values, and the convex hull, an example of a more complex type of community context calculation, were demonstrated in the first trial. In the second trial this component will be used as the basis for more complicated actions on community context, such as inheritance and prediction.
2.2.15 Context GUI (Luca)

Context GUI module is in charge of display, manipulate and create CSS data, that is stored in the Context Broker, in a graphical way. The Context GUI is web based and it is fully integrated inside the SOCIETIES web-app, using the JSF technology.

The module enables to access and display all Context CSS Entities, the list of Associations among them and the attributes that describes those elements. Here belo a screen shot of the Context GUI, available in the Profile Settings section of the Societies web-App.

**Figure 17 SOCIETIES web-app Context GUI fragment**

The GUI has an “Action Box Selector” that enables to perform all the Context Broker actions available:

- **Lookup** to perform Context queries by passing the Context Model and the Context Type
- **Retrieve** a single Context Element
- **Create** a new Context Element

**Figure 18 SOCIETIES web-app Context GUI fragment**

The “Result Table” below the box automatically displays the results of the action performed. It is possible to reduce the number of the showed entries by using the filters available in some of the table headers.

Each raw is editable by clicking in the **pen** icon. This means that is possible to change the value of the Context Element displayed and automatically store it on the Context Broker by the GUI.

The **Delete** icon deletes completely the raw in that table thus the Context Element associated in the Context Broker and all the attributes related to it (in the case of an Entity).

The Context GUI starts displaying all the Context Entities available for the logged User (the context Root tree). Some of them can be linked together by Context Associations.
Using the **Lens** icon, it is possible to navigate all the details of each Entity, displaying the associated Attributes and all the Context Associations available in another table. In the same way it is possible to navigate the Association (by the Lens icon) and see all the Entities involved in that Association.

<table>
<thead>
<tr>
<th>Id</th>
<th>Attribute</th>
<th>Value</th>
<th>Type</th>
</tr>
</thead>
<tbody>
<tr>
<td>14</td>
<td>sex</td>
<td>Male</td>
<td>MANUALLY_SET</td>
</tr>
<tr>
<td>19</td>
<td>nicknameReg</td>
<td></td>
<td>MANUALLY_SET</td>
</tr>
<tr>
<td>20</td>
<td>card_model</td>
<td></td>
<td>MANUALLY_SET</td>
</tr>
<tr>
<td>21</td>
<td>card_model</td>
<td></td>
<td>MANUALLY_SET</td>
</tr>
</tbody>
</table>

**Figure 19 SOCIETIES web-app Context GUI fragment**

The Retrieved Action enables to go straightforward to a specific stored element. For doing this it is necessary to know the Unique ID of the Context Element. Context GUI helps the user in this by providing a combo element that lists all the available identifiers.

**Figure 20 SOCIETIES web-app Context GUI fragment**

As mention before, Context GUI allows also to manually create new Context Elements and link them to the other Context Data available in the system.

### 2.3 Personalisation

The Personalisation Subsystem is responsible for the management and processing of all user behaviour models such as preferences, neural networks and intent models. Multiple approaches to modelling user behaviour have been utilised in the SOCIETIES personalisation subsystem. As each approach has its own strengths and weaknesses, the reasoning behind employing multiple systems was to ensure that in different situations, a different approach will provide the best personalisation technique. The performance of each technique is monitored and evaluated to indicate in which cases each approach performs better. As a result,
The personalisation and user agent subsystems will be able to apply the best personalisation practice in the corresponding situation.

The approaches used in SOCIETIES include:

- Dynamic Incremental Associative Neural NEtwork (DIANNE see ref 2.3.2).
- Context Dependent User and Community Preferences (see ref 2.3.3)
- Context Aware User and Community Intent (see ref 2.3.4)
- Conditional Random fields based uSer intenT (see ref 2.3.4)

![Figure 21. User personalisation component structure.](image)

The interfaces and classes in the common personalisation data model are used by multiple components in WP5. The IAction interface and Action implementation class is used to represent an action to be implemented by the T5.5 User Agent system. The action can be a command to be executed or a personalisable parameter to be set by a service. Both interface and class is exposed to external services and must be used by the services to record actions performed by the user and send these to the T5.5 User Action Monitor component to be used as input for learning user behaviour. The IOutcome interface extends the IAction interface and is only visible to SOCIETIES platform components. The IOutcome interface includes the getConfidenceLevel() that allows the Personalisation Manager and the T5.5 Decision Maker components to perform conflict resolution on multiple actions. The IOutcome interface must be implemented and, or extended by the each personalisation data model for their own use. The T5.5 User Agent component sends feedback based on the result of implementing an IAction object. The IFeedbackEvent interface and FeedbackEvent class is used by the T5.5 Decision Maker component to send feedback to the Personalisation Manager. The interface provides methods to retrieve the result of the implementation, the action the event
relates to, the identity of the user used in this service, the identity of the service provider and the service identifier. The FeedbackTypes enumeration provides a fixed list of implementation outcomes to indicate the reason for failure to implement an outcome.

![Diagram](image)

**Figure 22. Common Personalisation Data Model**

### 2.3.1 Personalisation Manager

The PersonalisationManager component orchestrates the prediction functionality of the following components: DIANNE, UserPreferenceManagement, CRISTUserIntentPrediction and CAUIPrediction. The Personalisation Manager registers for two types of events, context update events and user action events. It registers for action events upon initialisation but for context events, it provides a method that allows any of the above components to notify that they are interested in context events that signal changes in specific context attributes. Then it registers with the Context Broker for the events that notify changes to the specific context attribute. Upon receiving an event (user action or context), the Personalisation Manager requests the preference outcomes of the DIANNE and the User Preference Management components and the user intent predictions from the CAUI Prediction and CRIST Prediction components. The outcomes of DIANNE and User Preference Management are compared and resolved to reach non conflicting result and the same is performed for the predictions retrieved from the CRIST Prediction and CAUI Prediction components. The resolved outcomes are sent to the User Agent - Decision Maker component for implementation.

### 2.3.2 DIANNE

The Dynamic Incremental Associative Neural NEtwork (DIANNE) is a single layer, feed-forward neural network that learns associations between context and user behaviour in an incremental and temporal fashion. It is a real-time compliment to the offline batch preference learning algorithm (C4.5) and can respond rapidly to changes in user behaviour that indicate a change in user preferences. It is a continual process running in its own thread, constantly listening for context and behaviour updates and accommodating them into internal knowledge. Output is returned when requested by the Personalisation Manager.
2.3.3 User Preference Management

The User Preference Management component is an umbrella component for all the different functionality that is performed on the user preference models that includes managing, monitoring, evaluating and merging preferences.

The management part of this component handles the storage and retrieval of the user preferences to and from the context database which it uses as a storage medium. It maintains a registry of the user preferences that can be used to retrieve a user preference object directly from the database in order to avoid performing time consuming searches in the database. It maintains two caches for preferences and context values to reduce the number of calls to the context database which are updated appropriately by listening for context update events.

The merging part of the User Preference Management component listens for user action events and triggers a learning cycle when a counter has been reached. As soon as a new preference is learnt, the merging algorithm runs to merge the existing preference model with the newly learnt model. The result of the merging is stored in the context database.

The monitoring part of the User Preference Management is responsible for registering for context events with the Personalisation Manager for context attributes that affect the preferences of currently active services. When changes in the context of the user are pushed from the Personalisation Manager or changes to the preference models are pushed by the merging algorithm, a re-evaluation of the preferences based on the new input is performed and its outcome is returned to the Personalisation Manager.

Finally, the User Preference Management component is monitoring the user’s activities such as joining a CIS or sharing a service with a CIS. In the case of joining a CIS, the User Preference Management requests community preference information from the Community Preference Management component. It then uses the User Feedback component to ask the user which community preferences they want to download and merge with their own user preferences. Based on the response of the user, the component downloads the selected preferences and merges them with the existing user preferences. In the case of sharing a service with a CIS, the User Preference Management component will ask the user which of the preferences of the service they shared with the CIS they want to anonymously upload to the CIS. The selected preferences are then uploaded. In order to keep the user’s preferences and the community preferences updated, the User Preference Management component schedules uploading updated user preferences and downloading updated community preferences once a day.

2.3.3.1 User Preference Learning

The User Preference Learning component implements Quinlan's C4.5 decision tree learning algorithm. It operates on a request basis. When a request for a preference learning cycle is requested (by User Preference Management) an asynchronous thread is executed to retrieve behaviour history from context, process the history and extract preferences. The result is returned to requestor via asynchronous mechanisms.

2.3.3.2 User Preference GUI

The User Preference Graphical User Interface is provided by the SOCIETIES web-app component and allows the user to add a new preference and to view, edit or delete existing preferences. Context conditions can be added manually and they can be nested to form complex conditional expressions visualising them in tree structures. The leaves of the tree can only be preference outcomes (i.e. actions that indicate an action to be implemented) while the branches of the tree can only be context conditions.
2.3.3 Community Preference Management

Each CSS runs an instance of the Community Preference Management (CPM) component which is responsible for collecting user preferences from individual CSSs that are members of the CISs that the CSS owns. The Community Preference Management component is passive in its collection of user preferences. It exposes a remote interface which the User Preference Management component of member CSSs can use to upload user preference information to the CIS. Upon receiving a user preference from a CSS, the CPM checks to see if there is already a community preference for the same service and parameter. If it doesn’t exist, it will store the user preference as a community preference of that CIS. Otherwise, it will use the preference merging algorithm to enhance the existing community preference with the information contained in the uploaded user preference. In addition to the uploading method, the CPM exposes a method that allows CSSs to download community preferences from the CISs they are members of.

2.3.4 Context Aware User Intent - CAUI

The introduced CAUI model aims to describe the common behaviour patterns that a user or a community of users perform. CAUI is primarily describing actions that a user performs and possible sequences that arise among those actions. The often occurring sequences of actions are modelled as Tasks. For example when a user is about to start a driving trip is highly probable that will perform a sequence of actions, not necessarily with the same order each time. In this example CAUI will model the task labelled as “start driving” including the actions: “unlock car”, “regulate mirrors”, “turn on navigation service”, “set next destination”, “turn on car radio”, “set volume”, “turn on car engine” etc. In a similar manner possible patterns regarding sequences of Tasks are also modelled. For example the described task “start driving” will often be followed by the task labelled “end of driving” that will maintain all the necessary actions that the driver will perform in order to
stop and leave the car. Additional data regarding the context information that are related with user action and user task are also captured.

As it will be described the CAUI model is generic enough to model any kind of user action. However in terms of Societies we aim to model and predict actions that are related with use of services provided by the platform and third parties. The interactions among the user and a service will be captured by the Societies platform and stored in the Context History Database. The CAUIDiscovery component will then retrieve and process recorded data sets aiming to extract frequent occurring patterns. The CAUI Task Model Manager will model these patterns based on the CAUI model classes. Each Action is escorted with relative context information occurred on the same time.

![CAUI Class Model](image)

**Figure 24 Context-Aware User Intent (CAUI) Class Model.**

### 2.3.4.1 CAUI Task Manager

The CAUITaskManager component provides the necessary classes that allow the instantiation of user behaviour model. Additionally the component provides methods that allow the construction of the CAUI Model by creating and adding the necessary values to the appropriate UserIntentAction and UserIntentTask classes along with the respective transition probabilities among actions and tasks. Additionally this interface provides access to the CAUI model based on various criteria such as action or task ID, type, name.

### 2.3.4.2 CAUI Discovery

The CAUIDiscovery component is responsible for performing learning procedures aiming to create a user intent model. Upon request this component will retrieve data stored in Context History Database and based
on learning algorithms will extract common behaviour patterns of the user. The current implementation of the discovery algorithm calculates the number of transitions among subsequent actions and extracts the most frequent occurring patterns. In a similar manner context data escorting the actions is identified and maintained. The process results in a user intent model reflecting the most frequent user actions sequences and escorting context data.

### 2.3.4.3 CAUI Prediction

The CAUIPrediction component provides the necessary functionality for evaluating the CAUI behaviour model in order to predict the most probable User Action. The prediction algorithm considers the last performed action, the current context that describes user’s situation and previous prediction. Additionally the component triggers the user behaviour model discovery process.

### 2.3.4.4 Context Aware User Intent GUI

The Context Aware User Intent GUI allows the user to view and manage his/her personal intentions model which is learned based on his/her past actions. The GUI also allows the management of inherited community models which is used in combination with user’s personal prediction model in order provide secure predictions. The GUI also provides a log of performed user actions and respective predictions. Various screenshots of the GUI are depicted in the following figures.

In this figure the User Model tab of CAUI GUI is depicted. This part of the GUI supports the representation of the CSS owner’s CAUI model. The “Source Action” corresponds to a performed action and the “Target Action: transition probability” to the action that will probably follow.
In this figure the Action’s Context tab of CAUI GUI is depicted. This part of the GUI supports the representation of the prevailing context values that escort a user performed action.

In this figure the Community Model tab of CAUI GUI is depicted. This part of the GUI supports the representation of a CIS (community) CAUI model. The “Source Action” corresponds to a performed action and the “Target Action: transition probability” to the action that will probably follow. Options for selecting a CIS id, initiating a learning process and retrieving a fresh version of the CIS model are provided.
In this figure the Community Model tab of CAUI GUI is depicted. This part of the GUI supports the representation of a history log of user performed actions along with specific context values.

In this figure the Community Model tab of CAUI GUI is depicted. This part of the GUI allows the user to control (disable and enable) the overall functionality of the CAUI component.

### 2.3.4.5 Context Aware Community Intent (CACI) Discovery

The CACIDiscovery component is responsible for performing learning procedures aiming to create a community intent model. When learning is triggered this component contacts all contained CSS nodes that are part of a CIS and requests respective individual CAUI models. These models are processed in order to extract common characteristics and to build a common intent model that is appropriate for user action prediction. Finally the extracted model is distributed to individual CSSs and is used in conjunction with CAUI model aiming to maximise user prediction accuracy. The merging of individual models is based on a union operation of graphs.

### 2.3.5 Conditional Random fields based User intent

Conditional Random Fields (CRFs) are models for structured classification. In particular, CRFs are undirected graphical models, where the structure of the graph encodes independence relationships between labels. The term conditional appears in the name of the model because CRFs are probabilistic models that condition on the observations. In other words, the structure of the graph encodes independence relationships between labels and not the observations. The model assumes no independence relationships between
observations and, as a consequence, inference in CRFs remains tractable even when they incorporate complex and arbitrary features of the observations.

Conditional Random fields based User intent model (CRIST) aims to discover and predict user intents by leveraging Conditional Random Field (CRF). The CRIST Model is constructed based on three basic concepts, which are user action, user situation and user task. Specifically, we first obtain user actions and user situations through user action monitoring and user situation analysis; we model each user task as a sequence of user actions and the corresponding sequence of user situations; we build the relationship among user actions, user situations and user tasks by mining the historical data.

The CRIST information model consists of all the classes and interfaces for the construction of CRISTUserTaskModel and CRISTCommunityTaskModel. In each model, a list of user tasks will be modelled for future user action prediction and user task prediction. Particularly, there are four different classes and three different interfaces for each model, respectively.

![Figure 25 CRIST User Intent Class Model.](image)

**2.3.5.1 CRIST User Intent Discovery**

This component is responsible to construct the user intent model from an action history list. A record in the list is composed with an action and the accompanying situation. The intent model stores user action patterns and corresponding frequency. An intent model learning method will be called by CRIST User Intent Task Manager when intent model is needed, i.e., null or out of date.

- Available on Virgo.
- No need of communication.
2.3.5.2 CRIST User Intent Task Manager

This component is responsible for providing all necessary data structures and methods to maintain a prediction enabled environment. The data structures include an action history list, an intent model, a current user action, a current user situation, and a list of registered contexts. The methods called by CRIST User Intent Prediction include: predict user intent from current user context, current user action, and of certain service and parameter. They input current user action and current user situation, with the help of the intent model, outputs a list of possible actions. In order to implement the above functions, the intent model learning method of CRIST User Intent Discovery and some private methods are called, e.g., retrieve current user context and current user action from context broker, register context update to personalisation manager, and infer user situation from contexts.

- Available on Virgo.
- No need of communication.

2.3.5.3 CRIST User Intent Prediction

This component contains functions used by external services: prediction a user’s intent actions from current user context, current user action, and of certain service and parameter. They all depend on CRIST User Intent Task Manager. The basic idea is finding out in the user’s action history, which action is performed most of the time under certain situation and prior action.

- Available on Virgo.
- No need of communication.

2.3.5.4 CRIST Community Intent

This component is responsible for constructing the user intent model for a community (i.e., a group of users) from an action history list (consists of the action list of each individual member in the community and the action list when they orchestrate a community). Furthermore, in contrast, the CRIST intent model of a CIS could enhance the intent model of its individual member (CSS). Specifically, the three sub-components adopted in CRIST community intent discovery are as follow:

**CRIST Community Intent Prediction** — This component contains functions used by external services: prediction a community’s intent actions from current community context, current community action, and of certain service and parameter. They all depend on CRIST Community Intent Task Manager. The basic idea is finding out in the community’s action history, which action is performed most of the time under certain situation and prior action.

**CRIST Community Intent Task Manager** — This component is responsible for providing all necessary data structures and methods to maintain a prediction enabled environment. The data structures include an action history list, an intent model, a current user action, a current user situation, and a list of registered contexts. The methods called by CRIST User Intent Prediction include: predict community intent from current community context, current community action, and of certain service and parameter. They input current community action and current community situation, with the help of the intent model, outputs a list of possible actions. In order to implement the above functions, the intent model learning method of CRIST User Intent Discovery and some private methods are called, e.g., retrieve current community context and current community action from context broker, register context update to personalisation manager, and infer community situation from contexts.

**CRIST Community Intent Exploration** —This component is responsible to construct the community intent model from an action history list. A record in the list is composed with an action and the accompanying situation. The intent model stores community action patterns and corresponding frequency. An intent model learning method will be called by CRIST User Intent Task Manager when intent model is needed, i.e., null or out of date.

- Available on Virgo.
- No need of communication.

Community Intent Discovery Sequence Diagram (1. CSS1 create the CIS and create a CRIST community intent model in the meanwhile, and 2.CSS2 joins the CIS and updates its own intent model by using the community model)

2.3.6 Social Profiler

Social Profiler is a platform component that has been developed for dynamic management of user preferences and Behavior models.

- The prime scope of this component was to associate a percentage of pre-defined behavioral profiles related to specific interactions and actions performed in the social community.
- Actions and Interactions were analyzed from Facebook, Twitter, Foursquare and LinkedIn. All this data has been modeled by OpenSocial Social Data specification.
- Each profile percentage is calculated by periodically polling the user social interactions from his Social Networks and updating a graph (using Neo4J) that represents Friends, Groups and ActivityEntries nodes and interactions.

The module uses the Social Network component developed by WP4 that is able to access the data associated to the Social Network accounts linked with the Societies user. The SNS component normalized all the proprietary social data structure by using OpenSocial data specification. In this way all the data is threaten in the same way.
With this normalized data Social Profile generates a graph database using Neo4j where the element of the graph where compose by interests, General Info, Profiles, Persons, Places, Group and Pages. Neo Principles is base on Nodes, Relationship and Properties associated to both of them.

A Profile is the crucial Node that defines the behavior of a Person (the user plus all the other friends connector with him) based on its activities stream

A set of pre-configured Profiles is the first outcome of the Social Profiler analysis. Those profiles can be changed based on the context of the service, application or analysis that should be performed. In the case od Societies those Profiles where used to define the behavior of the user:

- **Narcissist**: it refers to the personality trait of self-esteem, which includes the set of character traits concerned with self-image or ego (e.s the user updates a lot his status or his profile)
- **Super-Active - Altruism**: is selfless concern for the welfare of others. The user replies frequently to his friends statuses, videos, applications, quizzes, photos+ comments or sends like-tags to his friends photos
- **Photo maniac**: uploads photos all the time and he/she tags people in his photos
- **Quiz maniac**: The user enjoys taking quizzes
- **Surf maniac**: the user interacted with the web, shares his preferences, you tube clips, links etc

Once the Graph is generated the **Eigenvector centrality** is a calculated, that means measuring the importance of a node in a network. It assigns relative scores to all nodes in the network based on the principle that connections to high-scoring nodes contribute more to the score of the node in question than equal connections to low-scoring nodes.

Here below the Data Model used to model the graph. A scheduler updated periodically the user information based on the activity Stream generated on the social networks. Once updates the Graph is traversed to update all the percentage associated to each profile. At the end the result of the analysis should be stored on the Context Broker in order to update the User Preferences and Information.
2.4 Privacy & Trust

The Privacy & Trust Protection layer provides components in order to protect the end-user’s personal data. It also aims to help the user understanding his personal exposition to the world, and taking the right decision about his protection.

The protection process is divided into several steps starting from a learning phase where the user privacy preferences are collected and the privacy policy for third party services and CIS are defined. Five steps can be described:

2. managing continuous learning of the user’s privacy preferences,
3. creating and managing privacy policies that will be associated to a CIS or a third party service,
4. negotiating an agreement between the privacy preferences of a user, and the privacy policy of a CIS or a third party service before to use them,
5. managing access control over personal data each time another user, CIS or third party service tries to access them,
6. collecting trust and assessment evidences to provide relevant feedbacks to the user.
The following diagram depicts the modules provided by the Privacy & Trust layer, both on the cloud node and the light node.

![Diagram of Privacy & Trust High-Level Architecture](image)

**Figure 27: Privacy & Trust High-Level Architecture**

As depicted, the “Privacy & Trust” architecture is divided into three main components: “Assessment”, “Privacy Protection” and “Trust Management & Evaluation”. Each component manages its own remote interface. This architecture provides a good modularity level, and avoids cyclic dependencies. In terms of implementation, the cloud node is based on Virgo; therefore these components are implemented as OSGi bundles for this platform. The light node is an Android application; therefore these components are packaged as Android services for this node.

In a more detailed point of view, each main component contains several sub-components, providing and using different APIs, in order to perform the whole protection process. These components can be depicted as bellow for the cloud node.
The architecture on the light and the cloud nodes are similar, even if the light version only performs a subset of functionality, and requires a running cloud node accessible through remote access. Therefore, the light node architecture is lighter than the previous one.

Figure 28: Privacy & Trust Cloud Node Detailed Architecture
2.4.1 Privacy Preference Management

The Privacy Preference Manager manages all the aspects of the privacy preferences of all types ("Privacy Policy Negotiation", "Access control preferences", "Identity Selection preferences" and "Data Obfuscation preferences"). The Privacy Preference Manager performs the following operations:

- **Storage and Retrieval.** Preferences are stored in the context database. The Privacy Preference Manager is responsible for indexing the preference objects and storing them accordingly in as context Attributes. To avoid multiple retrieval queries to the database, it also maintains a local cache of preference objects.

- **Proactive Preference Evaluation.** As Privacy Preferences can be context, trust or condition dependent, they are constantly monitored for changes. As soon as a context attribute or the trust value of a requestor that a privacy preference depends on changes, the Privacy Preference Manager re-evaluates the privacy preference against the current values. Depending on the type of the privacy preference, the Privacy Preference Manager will apply the result of the evaluation appropriately. In the case of an Access Control Preference, the Privacy Data Manager component will be used to update the current access control permissions. In the case of a Data Obfuscation preference, the Data Obfuscation component will be used to update the current obfuscation level that should apply for a type of data and specific requestor.

- **Behaviour Monitoring & Preference Merging.** When the Privacy Preference Manager receives a request to check the access permissions for a specific data type but there is no corresponding privacy preference to make the decision, it will use the User Feedback component provided by the T5.5 User Agent Subsystem to ask the user whether to allow or deny the request. The user input is then translated into an Access Control preference and merged with the existing preference if it exists. Depending on the type of data type requested, it is possible that data obfuscation can be performed on the data type. In this case, the User Feedback GUI allows the user to indicate if they want to obfuscate the data. This input is also translated into a Data Obfuscation preference and stored accordingly.

Figure 29: Privacy & Trust Light Node Detailed Architecture

The following paragraphs describe each component based on the cloud node architecture, and provide details about the light node when necessary.
Final state:

- Available on Virgo. No need on Android.
- No need of communication.
- Privacy Preference UI available on webapp (figure 15 below).

![Figure 30: Privacy Preference Editor](image)

2.4.2 Privacy Policy Negotiation Manager

Privacy Policy Negotiation (PPN) is the process in which two parties negotiate the terms and conditions under which they will interact and disclose data. There are two cases where privacy policy negotiation occurs; a) when a CSS wants to install a 3rd party service, the CSS will negotiate with the CSS that acts as a 3rd party service provider and b) when a CSS requests to join a CIS, it will negotiate with the CIS administrator. First, the privacy policy (“Request Policy”) of the service or CIS is retrieved and examined. The Privacy Preference Manager is used to evaluate all the PPN preferences related to the requests defined in the privacy policy of the service or CIS. Using the result of the PPN preference evaluation, the Privacy Policy Negotiation Manager generates a “Response Policy” in which the user defines his agreement or disagreement with each data request. This process involves the user consent, which is collected using the “User Feedback” component. The PPN GUI provided by the User Feedback component allows the user to customise a request. For example, the user can add additional conditions such as data retention or they can change the terms of an existing request such as changing the data retention period. After a successful negotiation, a negotiation agreement is created and signed by both parties and stored for auditing purposes as well as for access control purposes. This component provides the functionality that allows a CSS to act both as a client as well as a 3rd party service provider or CIS administrator. Depending on the role of the CSS during the Privacy Policy Negotiation process, the component will act accordingly.

Final state:

- Available on Virgo. No need on Android.
- User Feedback PPN GUI available on Webapp and Android.
2.4.3 Privacy Data Manager

The Privacy Data Manager performs access control over personal data from SOCIETIES each time another user, a CIS or a third party tries to access them. When such a request is received, the “Privacy Data Manager” uses the “Privacy Preferences Manager” to evaluate the most relevant preference according to this request and generate an access control result. This result is then stored using Hibernate, an object-relational mapping (ORM) library, in order to retrieve it quickly when the same request is received later.

If the access has been granted, and once the result has been retrieved, the Data Obfuscation Manager performs obfuscation on the data set in order to reduce its personal content. This is the second part of privacy access control, and it allows privacy protection to perform in very fine granularity. Each type of data requires a specific obfuscation algorithm even though some similarity and classifying can be determined. The Data Obfuscation Manager component provides the architecture to handle a specific algorithm for each data type, and methods to trigger evenly the obfuscation or the explanation to the user in a GUI. This component demonstrates obfuscation of two types of data: GPS coordinates, and user names.

To perform such access control and obfuscation, an integration process has to be done between the Privacy Data Manager and all SOCIETIES data providers. The SOCIETIES Context data are currently protected. The CIS Manager and the Social Network Connector are also using the privacy layer to preserve unwanted access to their managed data. To summarize, almost every SOCIETIES personal data are protected by the Privacy Data Manager. Only a few of them, not too critical in terms of privacy (e.g. Activity Feed), are not fully integrated with the privacy aspects. These gaps are not due to technical issues, but to project priority.

Final state:

- Available on Virgo (cloud node) and Android (light node).
- Remote access available.
- No need of specific UI.

2.4.4 Privacy Policy Manager

The Privacy Policy Manager is responsible for performing actions (creation, update, delete, read) under privacy policies and privacy agreements. These privacy policies will be associated to a new CIS or a new third party service, and the agreement documents will be securely stored after a privacy negotiation. These data are stored using the Context broker.

The privacy policy of a third party service lists the personal data needed by the service to complete its actions. The third party service developer creates this list using an external tool:
It is the same for a CIS privacy policy, except that the list of actions is defined by the SOCIETIES platform. The CIS creator only has to select a privacy policy between three choices: data available to nobody, to CIS members only, or to all SOCIETIES users.

Figure 32: Selection of a privacy policy during CIS creation - Android and Webbap UI
To finish another interface allows the end-user to visualize the privacy policy of a CIS or a third party service.

![Privacy Policy on Android](image)

**Figure 33: Privacy Policy on Android**

**Final state:**
- Available on Virgo (cloud node) and Android (light node).
- Remote access available.
- UI available.

### 2.4.5 Identity Selection Manager

The Identity Selection Manager selects the most relevant identity in terms of privacy. The SOCIETIES platform handles only one identity hence the current version of this component always returns the main identity of the user, support for multiple identities is not implemented. Because the privacy layer architecture and API implementing support of multiple identities should be almost transparent for the privacy layer. However, the impact in other SOCIETIES modules would require significant effort to be corrected and potentially delay more important project goals; that is why this module has been specified but no implemented.

**Final state:**
- Available on Virgo (cloud node) and Android (light node).
- No need of remote access.
- No need of specific UI.
2.4.6 Privacy Assessment Manager

The Privacy Assessment Manager component manages the assessment part of the privacy through two components: “Privacy Assessment Engine”, and “Privacy Logger”. The “Privacy Logger” logs events where private data is accessed or possibly shared or transmitted. The “Privacy Assessment Engine” parses the privacy log and assesses when, how often and which individual components and identities are possibly sending and receiving user’s private data.

The goal of such monitoring is to assess actual privacy practices. These might not be compliant with the promised privacy policies. And even when a disclosure of private data is compliant with privacy agreement, it may still not be in user’s best interest.

**Final state:**
- Available on Virgo (cloud node).
- No need of remote access.
- Specific UI available (see the chapter below).

2.4.7 Privacy Assessment GUI

The results of Privacy Assessment are exposed internally by Privacy Assessment Manager and displayed by its graphical user interface (GUI). The GUI is web based and integrated into the SOCIETIES Webapp. It aggregates numerical metrics retrieved from Privacy Assessment Manager and creates visual bar-charts from the metrics values. By viewing the charts, the end-user quickly gets the following information:

- which software components are accessing his personal data,
- which software components are transmitting data,
- which identities are possibly receiving the data, and
- estimation of correlation between data transmission and data access for each software component.

Some of the result types are shown in the following screenshots. The user can browse the results by clicking left and right arrows at the top left. If needed, the user can also limit the time interval of interest by setting start and end times.
Figure 34: Receivers of messages sent through the network.
While all these results help the user assess the actual privacy practices of the installed services, it is the correlation results that are most interesting, as well as most complex. For every SOCIETIES identity, OSGi bundle, and Java class that has ever transmitted any data, the correlation between the relevant data transmissions and data access is shown. The value of a blue or red bar in the figure below represents a measure of estimated possibility and frequency of transmission of local user data by a particular software component (SOCIETIES identity, OSGi bundle, or Java class). From the figure below, the user can see that:

- Five bundles have transmitted some data. Bundle names are shown, too.
- Three bundles (the ones with non-zero blue bars) could have in theory transmitted user’s personal data that has been gathered locally in the past from the context by any bundle.
- Two bundles (the ones with non-zero red bars) have accessed the user’s data themselves, and later transmitted some data.

The bundles are usually individual 3rd party services or SOCIETIES platform components. In case of 3rd party services, the service ID can be shown instead of bundle name.
Figure 36: Assessment of possible transmission of user data by the locally installed software.

Similar correlation bar-charts are shown for all classes and for all identities that have transmitted any data through the SOCIETIES platform.

Final state:

- Hosted on Virgo (cloud node), available locally and remotely through the web interface.

2.4.8 Direct, Indirect and User-perceived Trust Engine

The Direct Trust Engine is responsible for evaluating the trust evidence that result from direct interactions among the trustor (CSS owner) and the trusted entities (users, communities, or services), in order to estimate the trust level of the latter. More specifically, trust evidence includes information such as, trust ratings, social connections with other individuals, interactions with services, sharing/withholding context information, as well as, community membership data. After processing such information, the Direct Trust Engine is able to estimate the direct trust in the entities involved. However, trust assessment cannot be based solely on the experiences and evaluation of a user’s own interactions, but also on those of other trustworthy individuals. The Indirect Trust Engine employs collaborative filtering algorithms in order to make automatic predictions with regards to the trust evaluations of a user based on the trust opinions originating from that particular user’s trusted connections. The similarity of users with regards to their trust preferences is also taken into account during indirect trust evaluation, hence, the derived trust values are fully personalised. The User-perceived Trust Engine is then responsible for fusing the direct and indirect trust values of an entity in order to assess the aggregate value as perceived by the CSS owner. The direct trust value generally outweighs the indirect one in this fusion process. However, the weight of each factor also depends on the confidence level with which it has been estimated. For instance, when the direct trust evidence is not sufficient, the opinions from other CSSs have a greater effect in assessing the aggregate trust value.

Final state:

- Available on Virgo (cloud node).
- No need of remote access.
2.4.9 Trust Repository

The Trust Repository component provides a trust query interface on top of the underlying DBMS that actually controls the storage, management and retrieval of trust data. More specifically, this component is responsible for translating trust queries (w.r.t. direct, indirect and user-perceived trust information) into standard SQL queries that can be executed in the platform's DBMS. In this context, it utilises Hibernate which is an object-relational mapping (ORM) library, in order to map Trust Model objects, i.e. TrustedCsss, TrustedCiss and TrustedServices, to records in a traditional relational database.

Final state:
- Available on Virgo (cloud node).
- No need of remote access.

2.4.10 Trust Evidence Repository and Collector

The evaluation of direct and indirect trust in an entity is based on trust evidence. The Trust Evidence Collector is responsible for obtaining such information and storing it in the Trust Evidence Repository. This information can be of various forms and originate from various sources. More specifically, trust evidence includes locally collected data from direct interactions with services, CSSs and CISs (direct trust evidence), as well as, trust opinions from other CSSs (indirect trust evidence).

Final state:
- Available on Virgo (cloud node) and Android (light node).
- Intra-CSS and inter-CSS communication available.

2.4.11 Trust Broker and Trust Event Manager

The Trust Broker acts as a gateway to the trust calculations maintained either locally or in other CSSs. In this respect, it provides a query interface through which trust information consumers can specify various criteria, such as, the ID or the type of a particular entity in order to retrieve the evaluated direct, indirect or user-perceived trust values. It should be noted that trust queries can be performed either synchronously or asynchronously. In the latter case, the consumer can be notified upon trust value update events. The facilities required for subscribing and publishing trust-related events are provided by the Trust Event Manager. This component serves as an abstraction layer to the platform's underlying eventing mechanisms, i.e. WP4 Event Management and PubsubClient components.

Final state:
- Available on Virgo (cloud node) and Android (light node).
- Intra-CSS and inter-CSS communication available.

2.4.12 Trust GUI

A subset of the Trust Broker functionality is exposed through a GUI available for both the SOCIETIES Webapp (see Figure 37) and the Android app (see Figure 38). More specifically, this GUI allows the CSS owner to access the trust values evaluated by the system on their behalf. Assigning trust ratings to the users, communities or services evaluated by the system is also supported. On the SOCIETIES Webapp “Trust
Settings” page, the displayed results can be filtered by ID, while the user can specify a sort order based on the assessed value or time of evaluation.

**Final state:**

Available on Virgo (cloud node) and Android (light node).

![Figure 37 “Trust Settings” page in SOCIETIES Webapp](image)

![Figure 38. “Trust Level” information in SOCIETIES Android App.](image)
2.5 User Agent

The User Agent module is in charge of coordinating and managing the implementation of actions (i.e., the command to activate/deactivate a service or setting/resetting some parameters in a service) on behalf of the user, especially making decision for action implementation when the conflict existing between user’s intent and preference. User Agent consists of four key components—Decision Maker, Conflict Resolution, User Feedback and User Action Monitor; Specifically User Agent implements actions by incorporating these four components as follow:

- Given a list of intents and a list of preference, User Agent first uses Decision Maker to scan these two lists in order to detect the conflicts existing among the intents and preferences (i.e., two preference and intent actions set different value to the same attribute of the same service); and second the User Agent tries to resolve each of detected conflicts using Conflict Resolution component;
  - Given each detected conflict, the Conflict Resolution first tries to resolve the conflict by matching the predefined rules or the rules learned from the action data collected by User Action Monitor; If there exists no rules to resolve the conflict, the User Agent uses User Feedback component to query users in order to know the user’s choice and returns user’s choice as the result; Else if the conflict could be resolved, then Conflict Resolution returns the resolved result to the Decision Maker.

- Given the lists of intents and preferences without conflicts, the User Agent first builds a list of actions with conflict free; and Decision Maker implements the actions on the list sequentially, where, before implementing each action, it calls User Feedback to query user’s opinion with a timed pop-up (where the default setting is to confirm current action).

- Given the list of intents and preferences after conflict resolution, Decision Maker builds another list of actions; and Decision Maker implements the action on list sequentially, where, before implementing each action, Decision Maker calls User Feedback in following two ways:
  - If the confidence of the action is higher than the predefined threshold or is lower than the threshold but having been already chosen by user, then User Feedback to query user’s opinion with a timed pop-up, where the default setting is to confirm current action.
  - If the confidence of the action is lower than the predefined threshold and user hasn’t given any opinion on such action, then User Feedback to query user’s opinion with a pop asking to implement such action or not.

Rather than the invocations from User Agent, the component of User Feedback and User Action Monitor are widely used in the whole SOCIETIES platform. We will further elaborate the detailed responsibility of these four components in following sections.

2.5.1 Decision Maker

Given a list of intents and a list of preferences, this module is in charge of 1) detecting conflicts between preferences and intents, 2) querying Conflict Resolution in order to resolve all conflicts, and 3) implementing conflict-resolved actions using various levels of User Feedback. Further, this module collects user’s traces of action implementation (e.g., user’s opinion in solving the conflict, user’s choice before the action implementation) and forwards this information to Personalisation Manager in order to improve the performance of personalisation.

- Available on Virgo.
- Need to communication

2.5.2 Conflict Resolution

Given a pair of preference and intent having conflict (e.g., setting different values to the same parameter of the same service in the same time), this module tries to select on action on behalf of users according to the existing rules, where rules are predefined by users (e.g., the High-Confidence-Win rule) or are learned from
historical records provided by User Action Monitor. Of course, if there exists no rule to resolve a given conflict, Conflict Resolution will query user for user’s choice by incorporating with User Feedback.

- Available on Virgo.
- No need for communication

### 2.5.3 User Feedback

The User Feedback component provides interaction and notification mechanisms for use by platform components. It provides two modes of operation: explicit and implicit. Explicit mode requires interaction and feedback from the user before platform behaviour can continue. Implicit mode presents the user with a notification for a set amount of time. If the user does not reject the notification within this time period, it is taken as positive feedback and platform behaviour resumes accordingly.

![User Feedback Sequence Diagram](image)

**Figure 39 User Feedback Sequence Diagram**
Explicit notifications have been extended for the purpose of asking the user for informed consent during Privacy Policy Negotiations and Access Control Requests. Privacy Policy Negotiations occur when the user requests to join a CIS or attempts to installs a service shared by another CSS. The details of the negotiation are shown to the user who can then indicate whether they want to PERMIT or DENY access to the requested information, as well as being able to add or delete conditions for when that information should be disclosed. Access Control Requests occur when another CSS is requesting information about the current user. It presents a list of the requested data, allowing the user to choose whether to ALLOW or DENY access for each data type. The user can also select whether to remember their selection for each data type, so subsequent requests for that information can be disclosed/concealed without user intervention.

Figure 40 Android Login User Feedback Sequence Diagram
Figure 41. Privacy Policy Negotiation on Web-app

Figure 42. Access Control Request on Web-app
Notifications are accessed via the web app and Android app. The notification button on the menu bar of the web app highlights the amount of pending notifications in red, which is updated automatically as new notifications are issued. The Android app - unlike the web app which runs continuously – only receives notifications when the user is logged in. Therefore, each time the user logs in, the Android app sends a request to retrieve any unanswered notifications which can then be displayed to the user. While the user remains logged in the Android app, any subsequent notifications are pushed directly as they are issued.

2.5.4 User Action Monitor

The User Action Monitor (UAM) is responsible for collecting information from 3rd party services that indicate user behaviour. An interface is exposed in the SOCIETIES 3rd external API which services can use to send User Actions when users interact with the GUI of the 3rd party service. It is in the discretion of the 3rd party service to decide what kind of user behaviour they want to monitor and subsequently mined to produce user preferences or intent models. Upon receiving a user action, the UAM component retrieves a number of context attributes that describe the current environment and state of the user such as location, status, activity etc and attaches it in the form of a “Context Snapshot” to the User Action object. The enhanced user action now contains information that describes what happened and in which context it happened. The UAM stores it in the system using the Context History Management component. From there, the personalisation components can retrieve it as part of the user’s behaviour history. An action is also considered as a part of the context of the user and therefore, such information can affect the current personalisation evaluation outcomes. Thus, the UAM uses the Event Manager to publish a UI_Event to notify any interested components such as the personalisation components that the user has performed an action.
3 Prototype Evaluation

3.1 Lessons Learnt

This section provides a summary of the evaluation results from the student user trial at Heriot-Watt University and the enterprise trial at ICT 2013 conference. The full evaluation results and evaluation methodology can be found in the deliverable D8.9 (Final Evaluation Report).

3.1.1 Intelligent Orchestration

The goals of the SOCIETIES Relevance App can be divided into three parts:

1. From an attendee’s point of view to facilitate networking between attendees at the conference.
2. From an organiser’s point of view to gain access to statistics on session attendance post conference.
3. From a SOCIETIES project point of view to be able to validate a number of algorithms developed in T5.1. Namely;
   a. Collaboration Pattern Analyser
   b. Model Analyser
   c. Context Similarity Evaluation

![Prototype GUI wire frame layout](image)

**Figure 44. Prototype GUI wire frame layout**

**Attendees Perspective**
The scenario is to provide a Conference Organiser Application on Android for the ICT 2013 conference that expects in excess of 4,000 attendees. The application will perform the following functionality

- Allow users to register and provide profile details.
- Allow users to create their topics of interest keywords.
- Allow users to check-in at sessions.
- Allow users to tag sessions or topics of interest to them.
- Provide awareness of networking sessions that are active away from the main plenary type sessions that are of relevance to them.
- Provide the users with a conference schedule that highlights which sessions and speakers will be of most relevance to them.
- Provide users with exhibit lists of most relevance to them.
- Provide trending analysis on conference related Twitter from a conference perspective.
- Provide trending analysis on conference related Twitter from a specific session.
- Provide users the ability to create ad-hoc meetings and publish these to the conference community, with a request for relevant interested participants to join the meeting, such as collaborate on future proposals.

Attendee’s privacy is not at risk in this process as data is not shared between users or made available post conference. The App will unobtrusively recommend sessions and events to users based on their profile and conference activity only.

Post Conference Statistics available to Organisers

- Usage on actual numbers of people checking into different sessions.
- Visualisation of session and parallel session attendance per time segment and per conference.
- Visualisation of trending topics over the life of the conference.
- Ability to analyse voting statistics.
- Analyse people’s attendance against what they suggested they would attend during registrations.

Post Conference SOCIETIES analysis

The ability to validate algorithms from T5.1 namely

- Collaboration Pattern Analyser
- Model Analyser
- Context Similarity Evaluation

Another objective is to demonstrate scalability of a pervasive platform that can manage large numbers of attendees.

The modus of operation is to take a recommendation perspective and post conference to analyse what attendees selected to act on against the suggestions the SOCIETIES platform would have automatically recommended and engaged with.

Technology and Integration Discussion

The App needed to be scalable to handle the potential of thousands of users, expecting users to have the capability or desire to do this to run an App was unrealistic. The App team had to examine the touch points necessary from the platform required to demonstrate the algorithm functionality. To that end it was decided to implement new versions of some of the core technology.

- A new communication framework was required to manage the number of users and the number of potential messages required. This was implemented using PubNub.
A new context management dB was required as each user no longer had single containers with a unique DB per container.

**Results**

Due to poor network availability on the starting days of the conference and also due to the fact that there was an official App for the conference, take up of the prototype was poorer that had been hoped for.

- <50 downloads from the Google Play store.
- 21 concurrent connections was the peak usage over the 3 days.
  - Day 1, November 6th had 38 unique connections
  - Day 2, November 7th had 52 unique connections
  - Day 3, November 8th had 27 unique connections
- The server issued 48,986 App updates or relevance messages in one week of system testing and during the live event.

We can analyse some of the data, however due to the previously mentioned issues a broad set of user usage data is not available. In fact we only had two users who actively engaged over the course of the event with the App. We can see though that one of those registered with the following Keywords

- Internet of things.
- disaster recovery management.
- pervasive computing.
- social networking.
- big data.
- collective intelligence.
- context awareness

This pushed notifications to the user about 32 unique sessions that were happening over the 3 days, 2 sessions had a relevance score of 3, 1 session had a relevance score of 4 and the remaining sessions scores of 5. Later, during the conference this user tagged, voted and attended sessions that had been recommended to them by CSE. Can we definitely say that they wouldn’t have attended these sessions without prompting by CSE? Unfortunately not. Therefore, we are currently trying to find a more demo friendly conference that will allow us to become a more active participant and garner some more users and a wider interaction data set to work from post conference.
The team have since been approached to prepare this demo again for another EU conference in May 2014, which is seen as a positive endorsement of what the team tried to achieve during this demo. The work here has initiated an exploitation exercise, further details of which can be found here http://www.fuseami.com/.

3.1.2 Personalisation

The Heriot-Watt SOCIETIES platform included mechanisms for automated personalisation and for direct personalisation by the users.

Automated personalisation was used by three services, MyTV, the Calendar, and CoBrowsing. The automated personalisation process combined different learning mechanisms which it used to determine potential system actions. Four learning mechanisms were implemented for personalisation in the student trial: the DIANNE neural net, preference rules based on C4.5, CAUI and CRIST. Three of the mechanisms were fully integrated into the student trial platform and used by services - the DIANNE neural net, preference rules, and CAUI. CRIST was not fully integrated and only a very small amount of data from CRIST is included for completeness.

During the trial, an SQL database was maintained for each user, which stored:

- The DIANNE neural networks which were created for each user;
- The set of preference rules that were created for each user over the course of the trial;
- The set of CAUI rules that were created for each user over the course of the trial, and the actions that led to their creation.

The eventual learning from DIANNE neural networks was more limited in scope than had been planned. Each time the SOCIETIES platform was re-started the learning process started afresh with an empty network. As a result the networks did not accumulate knowledge continuously throughout the trial and the DIANNE network only demonstrated short-term learning rather than over the whole trial.
Preference rules, which were generated automatically, were visible to the users via the Web App and could be edited by the users. Figure 46 shows an example of a preference created automatically for the MyTv service, showing that the preference rule would propose different channels if the participant (User20) was logged in at the different screen locations.

Personalisation could be triggered by a change in context (e.g. time or location) or by the users’ actions. During personalisation, the outcomes from each learning mechanism were sent to a decision-making process, which chose which outcomes would be acted upon by the system. For the evaluation, the decision-making process was logged. The decision-maker first combined the outcomes from each of two different types of learning model; outcomes from the DIANNE neural net and the preference rules were combined into a set of “preference type” outcomes, while outcomes from the CAUI and CRIST models were combined into a set of “user intent model type” outcomes. Then the two sets of outcomes were combined into a single set of potential actions for execution. The personalisation mechanism could choose to execute its chosen outcome immediately, or it could first notify the user of its choice and allow the user to decide whether to execute the action. The decision-making process logged the triggers for personalisation (e.g. the users’ actions, or changes in context) and the outcomes that it received from each of the four learning mechanisms; it also logged the combination into preference type outcomes and user interaction model type outcomes.

![Figure 46 Example of a preference rule created automatically](image)

Personalisation: Acquired the following actions after receiving context event: locationSymbolic with value: screen3
1 dianne outcomes:
channel = 1
0 preference outcomes:
0 caui outcomes:
0 crist outcomes:
Out of the above, sending the following to the decisionMaker:
1 preference type outcomes:
channel = 1
0 user intent type outcomes:

The student is at screen3 and the DIANNE mechanism has proposed channel 1 at this location.

![Figure 47 Interpreted example of a log entry for automated personalisation](image)
Evidence of personalisation based on automated learning was found in the logs for six students. Table 1 shows the number of outcomes from each learning mechanism. This relatively low number may have several explanations. Only three of the six services were personalised in this way, and these were not frequently used. Thresholds were set before the mechanisms began to accumulate. In addition, DIANNE learning started afresh every two or three days so it did not accumulate enough data to operate frequently. Only four preference rules were created throughout, and two of these were created at the end of the trial during the SOCIETIES afternoon, which included a competition with MyTv specifically intended to exercise the learning mechanisms. Two students (User11 and User20) engaged in this competition and their logs show preference rule activity (Table 1). The two preference rules created by other students were incomplete and did not lead to any outcomes. Overall CAUI was the mechanism which gave most outcomes, as it accumulated data throughout the trial. The only user (User05) who had DIANNE outcomes but no CAUI outcomes had made little use of the personalised services.

### Table 1 Numbers of outcomes from each learning mechanism

<table>
<thead>
<tr>
<th></th>
<th>User01</th>
<th>User05</th>
<th>User10</th>
<th>User11</th>
<th>User15</th>
<th>User20</th>
<th>Total</th>
</tr>
</thead>
<tbody>
<tr>
<td>DIANNE</td>
<td>8</td>
<td>2</td>
<td>0</td>
<td>16</td>
<td>0</td>
<td>14</td>
<td>40</td>
</tr>
<tr>
<td>Preference rule</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>13</td>
<td>0</td>
<td>82</td>
<td>95</td>
</tr>
<tr>
<td>CAUI</td>
<td>21</td>
<td>0</td>
<td>264</td>
<td>274</td>
<td>22</td>
<td>96</td>
<td>677</td>
</tr>
<tr>
<td>CRIST</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>2</td>
<td>2</td>
<td>5</td>
</tr>
</tbody>
</table>

The personalisation process was triggered either by a context event (such as a change in location or time) or by a user intent event, typically an action by the user. Table 2 summarises the number of occasions on which the automated personalisation mechanism was triggered by each type of event. All types of learning could be triggered by changes in context or by user actions. For all of the users the great majority of automated personalisation was triggered by changes in context, i.e. time and location, even though most of the outcomes came from the user intent mechanism (Table 1). Some of this personalisation activity occurred due to clock changes overnight which were treated as context changes and led to personalisation activity although the activity had no observable effect since the students were generally not logged in at the time.

### Table 2 Numbers of personalisations triggered by changes in context and by user intent

<table>
<thead>
<tr>
<th></th>
<th>User01</th>
<th>User05</th>
<th>User10</th>
<th>User11</th>
<th>User15</th>
<th>User20</th>
<th>Total</th>
</tr>
</thead>
<tbody>
<tr>
<td>Context Events</td>
<td>21</td>
<td>2</td>
<td>263</td>
<td>278</td>
<td>24</td>
<td>169</td>
<td>757</td>
</tr>
<tr>
<td>User Intent</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>7</td>
<td>0</td>
<td>2</td>
<td>10</td>
</tr>
<tr>
<td>Total</td>
<td>21</td>
<td>2</td>
<td>264</td>
<td>285</td>
<td>24</td>
<td>171</td>
<td>767</td>
</tr>
</tbody>
</table>

Each learning mechanism creates a set of possible outcomes, from which the decision making process selects its actions. The different mechanisms generated varying numbers of possible outcomes (Table 3). The smallest number of outcomes generated by a learning mechanism is zero. The maximum number of choices generated varied for the different mechanisms, with the CAUI user intent mechanism generating larger numbers (8 maximum) each time. The DIANNE mechanism did not generate more than two possibilities and preference rules only generated one possibility each time the personalisation mechanism was used.

### Table 3 Maximum number of outcomes generated by each learning mechanism

<table>
<thead>
<tr>
<th></th>
<th>User01</th>
<th>User05</th>
<th>User10</th>
<th>User11</th>
<th>User15</th>
<th>User20</th>
<th>Max</th>
</tr>
</thead>
<tbody>
<tr>
<td>DIANNE</td>
<td>2</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>1</td>
<td>2</td>
</tr>
<tr>
<td>Preference rule</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>CAUI</td>
<td>0</td>
<td>0</td>
<td>7</td>
<td>8</td>
<td>1</td>
<td>8</td>
<td>8</td>
</tr>
<tr>
<td>CRIST</td>
<td>1</td>
<td>1</td>
<td>8</td>
<td>0</td>
<td>1</td>
<td>1</td>
<td>1</td>
</tr>
</tbody>
</table>
The eventual outcomes chosen by the decision maker were only partially logged. In the questionnaires, students gave a mixed response to the preferences that had been selected for them (Table 4). This mixed response was fairly consistent across the four weeks, with most students expressing themselves as “somewhat satisfied”.

**Table 4 Students’ overall perception of the preferences chosen for them (total over the four questionnaires)**

<table>
<thead>
<tr>
<th>Q2: Preferences</th>
<th>Not at all satisfied</th>
<th>Somewhat Satisfied</th>
<th>Satisfied</th>
<th>Very Satisfied</th>
<th>Delighted</th>
<th>N/A</th>
</tr>
</thead>
<tbody>
<tr>
<td>Q2: Preferences</td>
<td>4</td>
<td>12</td>
<td>7</td>
<td>5</td>
<td>1</td>
<td>7</td>
</tr>
</tbody>
</table>

The acceptability and value of the personalisation mechanisms were discussed during the interviews. Most students said that they had not used the three services frequently or extensively enough to confidently perceive learning, and the majority of trial participants stated that they did not feel they had experienced any learning. Twelve of the seventeen students interviewed didn’t observe any learning in the system. One student explicitly recognized experiencing learning with the MyTV system (User15). No one reported having experienced learning with the Calendar or CoBrowse.

Surprisingly some of those who did feel the system had learned something about them had interpreted learning in different ways from how the SOCIETIES researchers had intended, and attributed learning to services where it wasn’t available or other aspects of the system such as trust ratings or recommendations. Two students thought they had experienced learning in a service which did not have any learning - the Collaborative Quiz where students felt it was picking up on their habits and promoting their preferred categories (User10, User07). Other students thought they perceived some learning in relation to trust levels (User13, User01); while yet another student, interpreted learning to include recommendations, and said the mechanisms for learning or recommendations were not clear. (User20). These findings perhaps indicate that the students were anticipating, looking out for, and expecting to experience learning in the system and were perhaps disappointed not to have experienced more learning while participating in the trial.

- “I’d like to think so, but personally I don’t think it did.” (User17)
- "I didn't get the impression the system was learning about me. That's what I expected and that's what I thought was really good about it; that would be the innovation and the thing that gets interesting... but I don't think they actually did learn anything while I was using it." (User05)
- “I wish it could learn my GPS position so I could play GeoFence…” (User19)

One suggestion was that the recommendations offered by the system for making friends and joining communities, should learn from profile information to be more nuanced, as the small numbers in the group meant nearly all people were suggested as friends (User19). One participant noted that while they didn’t notice much auto learning they did appreciate the benefits provided by being able to change their preferences. (User12)

Few students expressed concern about learning, perhaps because few experienced it. Many indicated that they accepted the concept of learning in ICT systems. However a few made comments which hinted that their attitudes towards concerns about learning in the SOCIETIES system were formed from their general impressions about the pervasiveness of technologies in society at large, and the need for awareness in using such systems generally.

- “It would have concerned me, but in so far these days you can’t do anything without something... like your phone or your laptop picking up on what I’m doing. It’s one of those things when you use technology it will learn what your habits are.” (User09)
- “It didn’t really alter what I do. I’m always quite wary about what I do anyway.” (User07)
• This discussion lead to one woman, who accepted that the system was probably logging times and that the trial locations were standard, considering that the system should “Not [be] tracking me down because this is weird!” but could be useful for learning to track and remind her of approaching buses or whether other staff were available in their offices. (User19)

In addition to automated personalisation, students were able to personalise aspects of the system themselves. A mechanism was provided in the Web App for students to edit preferences explicitly, but few students did this. Only two students reported on the questionnaires having tried to change their preferences.

### 3.1.3 Privacy

Privacy was a significant concern raised by participants in the earlier trial and therefore privacy mechanisms were an important issue for the design of the platform. The platform provides explicit mechanisms for its users to manage the privacy of the data that they share with other users and with services. An initial privacy negotiation is done at service installation and on joining a CIS. Privacy preferences could also be inspected and changed manually at a later date. When the trial managers installed each service, they went in to each student’s account and conducted the privacy negotiation on their behalf. These negotiations were always successful, showing that the mechanisms worked well. Students were responsible for their own initial privacy negotiations when joining a CIS. They could also inspect their privacy settings and change them at any time using the Web App.

During the interviews, while several students answered “no” initially to the question about whether they felt SOCIETIES was trying to protect their privacy, most agreed that they were aware of the notifications. One student answered “not really” but reported noticing a preference list for access controls. Some felt that the language in the notifications was unclear, (User15). Another felt that the preferences (in general) were confusing and complex to use. This participant would have preferred a more simple privacy interface, as “socialising with friends should not be a chore” (User13). This would seem to indicate that many of the students might have experienced receiving the notifications during the use of the system as a sufficient sign that their privacy was adequately protected, without feeling the need to actually read the notification texts more closely.

• “I gave that straight away. I never read that... In smart phones nowadays everything requests permission so it’s like a straightaway thing to do” (User12 gesticulating swiping acceptance, while she had read the initial privacy terms and conditions carefully.)

Notifications were mentioned by several participants as a usability issue when using the application. Although some students reported that there was scope for improvement in the privacy interfaces, many reported seeing no need to intervene in their own privacy negotiations in the context of the trial. The number of notifications was also perceived as an annoyance by some but a necessary one by many as evidenced in the answers related to requests for permissions before initiating the pervasive screen services.

Both the log analysis and the interviews showed that students did not change their privacy settings. While a few students Users 17, 20, 15, 10), remembered that they had at least looked at the privacy settings initially, to check them, most said that they didn’t feel any need to alter them then or subsequently.. More didn’t know that they could, either because they didn’t use the Web App at all or didn’t notice or feel the need to use that functionality. This was mostly because they were sufficiently satisfied with the default settings, and also because they didn’t think the system was capturing any valuable personal information, and they did not think they were sharing any private or personal data, so they did not put any effort into protecting it. In addition at least one participant (User17) stated that the information that was given during the trial induction process provided sufficient assurances about how the data gathered during the trial would be managed:

• “No but the information it gathered on me was not that valuable, for me to care as such... and also it was stated at the first meeting about what information would be gathered.” (User17)
• “The system didn't seem to be learning anything about me that I had a problem sharing.” (User11)
• “I did feel whenever it wanted more information from me it would ask if it could be approved.”(User20)
Few interviewees thought they changed privacy preferences, one person thought s/he may have done by accident (User08) while another was trying to make his privacy preferences more open (User10) and did so by changing the settings to display his real name rather than a username.

- “I think the privacy levels were pretty balanced. You had quite a few levels you could have applied... It all depends on what you are willing to put into your use. Lots of people use systems without putting in legitimate information.” (User10)

The consideration that people might use incorrect or placeholder information was also discussed in the interview with User15.

Most users admitted being unaware that the functionality to create new privacy policies existed in the system, and didn’t think it was necessary for the trial environment. Even those few students who had discovered it did not feel the need to create new privacy policies.

- “I think I saw that functionality somewhere.. but I didn’t think my privacy preferences needed to be changed at all.” (User20)

No students felt the need to create new privacy settings, partially because it seemed complicated but more students just stated that they didn’t need to.

- “I don’t think I would have understood it. It looked a bit complicated in the induction. Google circles is as far as I would like my privacy to go is the settings I would like....” (User13)

Some made distinctions between the trial and “real life” noting they might want more protection in ‘real world’ scenarios. One participant stated that his default was to disclose the minimum and his expectation was that all information shared with social networking systems could be in the public domain. (User10).

Another student stated that while: “for the purpose of the trial, even if I knew it, I wouldn’t do it because I want to help you log stuff (data for the trial)” (User19); but later went on to say that SOCIETIES services appeared to be very separate, but if for example GeoFences and Calendar were closely related and s/he was on her way to a conference s/he might have checked out creating a particular privacy policy. This indicates that management of privacy policies is a complex task for users, and while they were complacent within the safe confines of the trial environment, some were aware of a need to be more vigilant about their privacy behaviours in the wider world, and would appreciate the facilities offered by the SOCIETIES trial platform to tailor their privacy preferences, at least in some particular situations.

Overall, students were satisfied with the privacy settings provided within the scope of the trial and privacy was not a barrier to their use of the system.

3.1.4 Trust Management & Evaluation

The prototype implementation of the Trust Management & Evaluation system has been evaluated through a series of trials by all three SOCIETES user groups. These user trials were conducted in realistic environments and, despite the prototype nature of the implementation and the occasionally restricted number of participants, the analysis of the recorded data along with the feedback captured, indicate that the Trust Management & Evaluation system is able to perform well in a variety of CSS usage situations.

Based on the data collected during the six-week Student trial conducted at HWU, we have extracted useful information regarding the established trust relationships and the acquired trust evidence. The results are presented hereafter.

Trust Relationships established during the Student Trial

Figure 48 depicts a graph in which the CSS of each trial participant is represented by a node, while trust relationships are represented by directed edges between trustor-trustee pairs. These relationships, depending on the estimated user-perceived trust value, have been classified as strong, neutral and weak. As such, graph edges have been coloured green, black, and red, respectively. The 29 strong trust relationships that formed during the Student trial are illustrated in Figure 49. As expected, both neutral (cf. Figure 50) and weak (cf. Figure 51) trust relationships far outnumber strong ones, totalling 148 and 108, respectively.
Besides other CSSs, users also established trust relationships with CISs and services. More specifically, analysis of the Student trial data shows that users have formed, on average, 1.208 strong trust relationships with CSSs, 2.125 with CISs and 1.083 with services. The results of this statistical analysis for all types of trust relationships have been included in the following table:

<table>
<thead>
<tr>
<th>Trust relationship type</th>
<th>Trusted entity type</th>
<th>Average number of trusted entities</th>
<th>Standard deviation</th>
</tr>
</thead>
<tbody>
<tr>
<td>Strong</td>
<td>CSS</td>
<td>1.208</td>
<td>2.081</td>
</tr>
<tr>
<td>Neutral</td>
<td>CSS</td>
<td>3.875</td>
<td>3.689</td>
</tr>
<tr>
<td>Weak</td>
<td>CSS</td>
<td>4.5</td>
<td>3.905</td>
</tr>
<tr>
<td>Strong</td>
<td>CIS</td>
<td>2.125</td>
<td>3.257</td>
</tr>
<tr>
<td>Neutral</td>
<td>CIS</td>
<td>9.167</td>
<td>8.454</td>
</tr>
<tr>
<td>Weak</td>
<td>CIS</td>
<td>6.833</td>
<td>4.497</td>
</tr>
<tr>
<td>Strong</td>
<td>Service</td>
<td>1.083</td>
<td>0.64</td>
</tr>
<tr>
<td>Neutral</td>
<td>Service</td>
<td>4.5</td>
<td>1.118</td>
</tr>
<tr>
<td>Weak</td>
<td>Service</td>
<td>18.75</td>
<td>16.184</td>
</tr>
</tbody>
</table>
Figure 48: Social graph of Student trial participants. The CSS of each participant is represented by a node, while the established trust relationship is represented by an edge. Depending on the estimated user-perceived trust value, we distinguish among strong, ne
Figure 49: Strong CSS trust relationships established during the Student trial.

Figure 50: Neutral CSS trust relationships established during the Student trial.
Figure 51: Weak CSS trust relationships established during the Student trial.

Trust evidence collected during the trial

The minimum number of trust evidence collected during the Student trial is 71, while the maximum is 126895, which indicates a great variance. On average, the CSS of each participant gathered 12152.25 pieces of trust evidence with a standard deviation of 27035.91 and a median value of 889.5. Due to the skewness of the observed numbers of trust evidence collected during the Student trial we have used logarithmic scale to plot them in Figure 52.
Figure 52: Number of trust evidence (in logarithmic scale) collected per CSS during the Student trial.

It should be emphasised that out of the 291654 recorded trust evidence during the Student trial, only 17256 were user-specified trust ratings. This amounts to 5.92% of all trust evidence or 16.86%, on average, per CSS. However, most trust evaluation systems assume that trust ratings of resources are available, whereby, users are expected to rate items, e.g., through like/dislike or star classification schemes. Relying on a rating system that is averaged across all users aiming at a global trust calculus (reputation) cannot be personalised, and is particularly poor in tasks where there is large variation in the items of interest, as in the case of CSSs where the available resources span from individuals to communities and a multitude of services. Contrary to the artificial behaviour imposed by rating-based systems, trust evaluation in SOCIETIES builds upon unobtrusive observations of actual user behaviour by means of the pervasive and social computing facilities of the platform. Thus, while the trial was restricted to a limited number of participants and services, thereby narrowing the possibility of a genuine network effect, the system was still able to infer trust relationships among users and their interacting entities, which is a real achievement.

3.2 Future Directions

3.2.1 Intelligent Orchestration

The team are currently trying to find a more demo friendly conference that will allow us to become a more active participant and garner some more users and a wider interaction data set to work from post conference. The team have since been approached to prepare this demo again for another EU conference in May 2014, which is seen as a positive endorsement of what the team tried to achieve during the Relevance demo. The work here has initiated an exploitation exercise further details of which can be found here http://www.fuseami.com/.
fuseami is a back-end as a service (BaaS) platform for mobile application developers to enable them to create dynamic social and community based apps. Fuseami stands for the Fusion of Geo, Social, and Ambient Intelligence capabilities, as a service, to enable developers to easily create applications that include smart geo-fenced environments or dynamic social communities on the fly.

Fuseami also provides developers with an advanced “Similarity Engine” that can be used to identity commonalities between people based on interests and locations, providing a variety of relevance based services.

### 3.2.2 Context

Potential future directions to be investigated and plans regarding the Context Management components implemented in SOCIETIES include the following:

- Integration of the database model used in SOCIETIES to a NoSQL database, which performs better in cloud platforms.
- Extension of the context data collected to other sources, such as other social media channels (e.g., Instagram, flickr, xing, etc.) and additional sensors.
- Enhancement of the accuracy of the location estimated by IBM’s Presence Zone Server (PZS), by employing heuristics and exploiting user feedback.
- Extension of the context inference mechanisms (regarding context refinement and context prediction) with more techniques and algorithms inspired from fuzzy logic, game theory, stochastic process theory.
- Attempt to perform multi-dimensional and fully quantifiable context similarity evaluation, even for non-numerical context values, to compare the similarity of current, previous or predicted situations for: two users, one user and a specific community, two communities.
- Further evaluation of the user and community prediction and refinement mechanism developed in SOCIETIES via larger and more complete context datasets, covering more context information types monitored for about 1000 users for a period of some months. In this respect, not only the accuracy of the inference mechanisms will be evaluated, but also the performance of the entire CSM regarding for example query processing, training, scalability, etc.
- Development of a more user friendly and illustrative GUI by also employing visual analytics to properly visualise the context associations that may exist among context entities, including CSSs, CISs, Persons, Services, Resources, Objects, etc., as well as by extending the context query support with more search criteria.

### 3.2.3 Personalisation

From the data collected during the conducted user trials, it was observed that different personalisation techniques worked better in different situations proving that employing multiple personalisation techniques was a sound decision. Based on this finding, we intend to investigate this further with the purpose of providing a holistic approach to personalisation. We plan to conduct further testing, evaluation and further user trials focused on extracting more data to support this theory.

We also want to conduct more research on using user preferences for personalisation. Specifically two major issues need to be addressed: a) the issue of visualising the user preferences to make them easier for users and b) treating user actions as part of context so that this information can be considered conditional to further outcomes. The latter issue also involves researching the best way to model user behaviour information in the same form as context.
3.2.4 Privacy Preferences

Based on the data collected and the experience during the student user trial, we have identified a number of areas that need further investigating. We plan to further evaluate the use of all different types of privacy preferences in the privacy protection cycle aiming to build tools to educate and alert the users about data disclosure practices. Specifically, our objective will be to investigate the use of privacy policy negotiation preferences in cases where the negotiating party (CIS administrator or 3rd party service provider) is unknown to the user. Thus, suggesting several alternatives to the user based on a) the user’s disclosure behaviour with other similar requestors, c) the trustworthiness of the requestor in the community and c) the sensitivity of the data to be disclosed. Furthermore, existing graphical user interfaces have to be re-examined in order to educate and provide a clearer picture to the user about what the privacy preferences are used for.

3.2.5 Trust Management & Evaluation

We plan to further evaluate and test the prototyped Trust Management & Evaluation system aiming to identify any scalability problems that may arise, especially in cases where the number of users considered in the indirect trust evaluation process is very large and the number, nature, experience and rating of their respective interactions greatly vary across time. Furthermore, it is planned to extend this system to support community-assisted trust learning and prediction to be applied for entities that have minimal interactions with users and communities. Finally, we plan to introduce uncertainty in the trust model designed and experiment with stochastic modelling of the various trust formulas used.
4 Conclusion (HWU)

The CSS Individual and Community Experience of the SOCIETIES platform provides innovative practices for combining pervasive computing with social interactions such as intelligent community orchestration, context aware dynamic and proactive personalisation, personalised and dynamic privacy protection, user and community context awareness and finally intelligent decision making and user feedback collection.

The components that provide the CSS Individual and Community Experience are constantly monitoring the user’s environment and activities, discovering appropriate resources, services and communities and evaluating their usefulness for the user. Intelligent community orchestration takes advantage of several WP5 innovations such as context, personalisation and trust to suggest to the user the creation of new communities or joining existing communities. Personalisation reacts to changes in the user’s environment and new interactions of the user with other users, communities, services or resources to adapt both the physical and digital space that surrounds them. Context, personalisation and community information are protected using innovative privacy protection technologies such as personalised privacy policy negotiation, access control and data obfuscation.

This deliverable summarised the architecture of the CSS Individual and Community Experience of the SOCIETIES platform implemented in work package 5 (WP5) and gave an overview of the evaluation results of the data collected during the user trials conducted for SOCIETIES.