Applied Mathematics & Information Sciences
An International Journal

http://dx.doi.org/10.12785/amis/070604

An End-user Tailorable Generic Framework for Privacy-preserving Location-based Mobile Applications

Dhiah el Diehn I. Abou-Tair[1,*], *Mohamed Bourimi*[2], *Ricardo Tesoriero*[3], *Marcel Heupel*[2], *Dogan Kesdogan*[2] *and Bernd Ueberschär*[4]

[1] School of Informatics and Computing, German Jordanian University, Amman, Jordan
[2] Chair for IT Security, Privacy, and Trust of the University of Siegen, Siegen, Germany
[3] Computing Systems Department, University of Castilla-La Mancha, Albacete, Spain
[4] GMA Association for Marine Aquaculture, Hafentörn 3, 25761 Büsum, Germany

Received: 15 Mar. 2013, Revised: 18 Jul. 2013, Accepted: 19 Jul. 2013
Published online: 1 Nov. 2013

Abstract: In this paper, we discuss the emerging need for supporting end-user tailorability and privacy with respect to collaborative location-based applications in mobile environments. We present a generic privacy-preserving framework for supporting such end-user tailorability on both user interface and server-side ends. The requirements were compiled from various use cases of collaborative location-based scenarios from various projects and related literature. The outcome results of this work demonstrates the feasibility of our proposed framework by means of an iOS based prototype applied to previous iAngle and iFishWatcher research projects. The significance of this prototype that it allows different communities to define their own points of interest in a generic manner. Additionally, the distributed architecture can be tailored according to user privacy requirements.

Keywords: Location-aware applications, end-user tailorability, mobile computing, location privacy, location-based services

1 Introduction

Mobility aspects are becoming more important than ever and affect many directions of our professional as well as leisure lives. This includes also academic research, since many communities leverage the usage of mobile devices in order to identify endangered animals [10] or to compile occurrence records of existing species[1]. Latter is the case in our on-going project concerned with the development of mobile applications for angling communities to support sustainable management of aquatic resources and biodiversity research (see below). By this means, location context information plays a key role in used mobile applications, e.g. in defining community-specific points of interest (POI).

In general, location-aware applications are experiencing a widespread usage in our private life (e.g. Foursquare, Google Latitude, etc.). As location and mobile computing converge in different form of applications, it becomes a need to study location-based/-aware applications and services anew. Especially in our new digital age such a study should be with emphasis on privacy, then privacy is the most-cited criticism for mobile, pervasive and ubiquitous computing [11]. Further, the way in which existing applications and services are constructed show the need to a generic framework that supports end-user tailorability as we observed in our case.

This work aims at constructing a generic framework for location-based mobile applications and services. The focus is to support end-user tailorability, by combining two conceptual approaches and a corresponding user interface (UI). The UI supports end-users in defining and managing their own POI by using their preferred distributed architecture according to their privacy preference. With this, the same framework can be re-used by different communities (not just our angling community) by adapting/tailoring it to their specific POI and related categories that can be created at-runtime and this in a flexible and easy way.

The remainder of this paper is structured as follows. The next two sections discuss background information

[1] http://www.wiwi.uni-siegen.de/itsec/projekte/iangle/index.html

* Corresponding author e-mail: dhiah.aboutair@gju.edu.jo

and address the phase of gathering requirements. Section IV compares our work to related work while section V presents our conceptual approach. Section VI discusses the implementation of the prototype and the paper is concluded in section VII.

2 Projects' Background Information

The end-user tailorability and privacy needs we address in this work were identified in the ongoing work of the inter-disciplinary iAngle and iFishWatcher projects at our University and collaborating institutes. The iAngle project itself emerged as a spin-off from the EU project PICOS[2], which deals with privacy and identity management in mobile communities. PICOS followed a user centered and scenario based proceeding for eliciting the gathered needs and requirements (such as user experience, interviews and questionnaires) for two different mobile communities (the Angling Community and the Gamer Community). The Angling Community is built by recreational anglers who explore water bodies and coastal areas, even the unproductive and those inaccessible to commercial fisheries, to an extent that is unattainable by scientific projects. They spend enormous time and effort investigating fish communities. In recent years there has been much effort, through initiatives such as OBIS[3] and GBIF[4] , to compile occurrence records of existing species, and to make these records available online in a standardized format, including the distribution of finfish on a global scale (see Figure 1). Such occurrence records, derived from surveys and inventories, provide the essential baseline data for monitoring changes caused by factors, such as human usage of biological resources (e.g. fisheries) habitat conversion, climate change, and help in determining in-country priorities on species conservation. There is no doubt that information on global species diversity is necessary to support well-informed decision making at the global level, yet information critical to such decisions are not available readily [8].

In order to encourage angler to share their data for scientific examination, FishBase, the global online information system about fish, established the FishWatcher facility several years ago. FishWatcher is a place where angler and other fish watcher can upload their catch data and observations. However, the numbers of records shared has been disappointing over the lifetime of the project. The lack of willingness exhibited could have been that their records would become visible to the entire community, i.e. anglers did not consider uploading their data simply because they did not want to disclose it or attract commercial fishing boats to their favourite fishing spots. Nevertheless, the call for improved

2 http://www.picos-project.eu

3 http://www.iobis.org

4 http://www.gbif.de

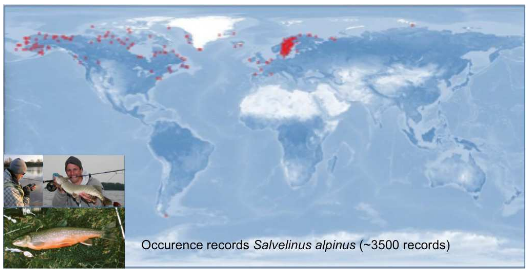


Fig. 1 Occurrence records by fish watcher

communication and better understanding of the different perspectives among fishery scientists, managers and the recreational fishing sector has been identified as a pressing need in recreational fisheries management throughout the world [5]. Our innovative approach to improve recreational angler participation in science and research makes use of the latest smart phone technologies. We have introduced two new mobile angler applications for the iPhone and a client for Android smartphones providing features specifically designed for the angling community (this paper, however, deals merely with the iPhone client). The major objectives of our work are to support the mobilization of the catch record data from the angling community for scientific purposes. The iOS-based iFishWatcher mobile application is a catch diary and a social community client for iOS devices, such as iPhone, iPad or iPod Touch. It covers two main focus areas, namely, management of user catches and interaction among members of the mobile angling community using a secure and decentralised architecture, which could be also used in a centralised way. To achieve catch management functionality, the iFishWatcher application makes use of two different resources: (1) the FishBase database, which provides rich information about fish since the user is actively supported in creating new catch diary entries based on the scientific information in FishBase[5] and (2) the iFishWatcher server.

In a catch report, the user typically provides the common name, physical attributes (size and weight) and takes a picture of the catch that is saved on the device and in cases where the angler wants to share his catch record, uploaded to the server. The catch information, including the geo-location of the fishing spot, obtained by the devices sensor, can be shared either with a limited number of buddies or the entire community (see Figure 2) Since this may result into privacy respecting concerns, we have also included the possibility to set the location manually in order to blur the user's actual position. The second resource, the iFishWatcher Server, provides online access for management of personal profiles, catch diaries and the collaborative mobile Location-Based Services (LBS), e.g. the fishing spot facility. Users may run their

5 http://www.fishbase.org/home.htm

own server in their respective community so that they are able to administer transmission data on their own (members building a trustworthy sub-community) or just use a global instance of the server that is managed by a third party. In order to support the aforementioned features, the server makes use of three key technologies as described in [2]:

1. The Spring and the Spring Roo framework that is providing support for all user profile and catch diary related tasks and automatically creates Web-Interfaces for the basic operations that help user to manage his data online using the browser of his choice.
2. RESTful, a web services capable of sending and receiving JSON formatted data over HTTPS. This way we ensure that encrypted data is transferred with the least possible amount of overhead that is an important issue regarding privacy, security and usability, since it is reducing response times by far and thus makes the App usable even in areas with bad signal strength and low speed internet connectivity. Using these services, the user can upload locally saved catches to the server or download all of his catches that are present on the server what enables multi device usage.
3. The third key technology is the XMPP framework, necessary for collaborative scenarios (see chat example in Figure 3 a). Users register an account with a server and are able to use multiple accounts with different servers sharing data with the respective community (e.g. removing catches in Figure 3 b). Utilizing the XMPP protocol, one is able to provide chat functionality as well as catch and geo-location data exchange between users who added each other to so called buddy lists (see Figure 3 c). Users can state an individual privacy level by defining buddy-by-buddy rules that restrict access to their location and online status. If data exchange is permitted, users can see the catches of their friends and can track a users location, as well as the location of their friends fishing spots, using the built-in Google Maps facility.

The main difference points between PICOS and iAngle are represented in Table 1.

Further information about the iFishWatcher project and related research (e.g., enhancements with respect to privacy advisory on mobile devices described in [3]) can be found at the website www.ifishwatcher.org (iAngle demo videos and up-coming functions).

3 Problem and Requirements Analysis

Early, in the design and implementation phases of the iAngle prototype it became clear, that a generic solution supporting different kinds of POI could reduce development costs. The same observation was experienced while building the 2nd PICOS Community

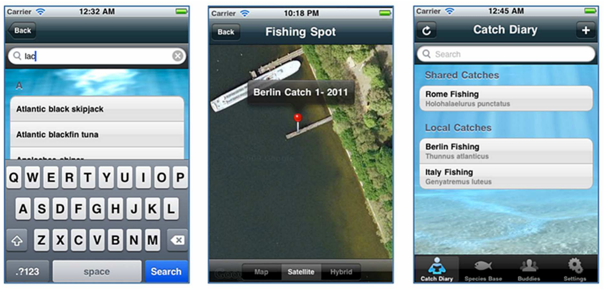


Fig. 2 Selected catch functionality in iFishWatcher

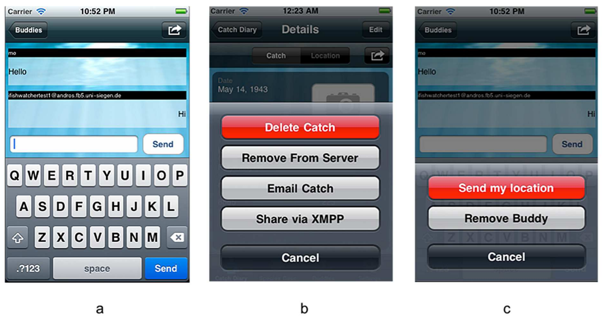


Fig. 3 Selected collaboration examples in iFishWatcher

Prototype [16], which addressed the requirements of the Gamer community. Server-side functionality could be re-used when generic POI are supported. While anglers are interested in watercourses and fishing spots, gamers want to see Internet cafe locations and WiFi spots on their mobile clients. Manipulation functionalities such as adding, removing or editing these POI stay the same for different communities. However, only the members of a given community could appropriately define their (context-related) POI. Non-angling people for their own target scenarios (teaching staff and students at our University who tested the iAngle prototype) also explicitly requested this. For this purpose, we identified the requirement *to support end-user tailorability with respect to different POI with a generic mobile framework by considering the usability of providing it (High-level requirement 1 - HLR1).*

The second HLR is taken from the iAngle and iFishWatcher projects. This project is part of the FishWatcher[6] project based on the previous iAngle project. A requirement for supporting different community schemes was identified. For instance, the users asked for having a global data server that could be used by all angler communities in different countries

[6] iFishWatcher: http://ifishwatcher.org/

Table 1 Comparaison between iAngle and PICOS

Aspect	PICOS	iAngle
Architecture	Server-centric	Allows for decentralization
Server-side & Software	Subset of HP Open Call (PHP and not available!)	Spring-based server
Anonymous usage	Signing in requested	**passive** usage possible
Identity Management	Root identity + various partial identities	Simple pseudonyms **(Community requested!)**
Mobile Client & Software	Nokia Music Xpress running with Java on Symbian OS	iPhone SDK 3.x
Simple navigation & improvements related to entering data	Limited interactivity due to Java	Good support of iPhone SDK

whereas each country provides just a local communication and awareness server for the members. Since users of the same community differently interpret privacy, we identified the requirement *of enhancing privacy by enabling end-user tailorability of the communication and data sharing schemes they intend (High-level requirement 2 - HLR2).*

4 Comparison to Related Work

One way of reacting on emerging changes is to allow for tailorability. Henderson defined tailoring as *”the technical and human art of modifying the functionality of technology while the technology is in use in the field”* [9]. Bannon argues that *”there will always be a need for some form of tailoring in order to fit a system into any particular setting”*[1]. When developing socio-technical systems and applications, satisfying user needs and requirements is even more difficult than in the context of single-user application development. To achieve the task-technology fit[7][18], different approaches suggest different levels of tailorability. Thereby, the tailorability level varies from supporting customisation, extension or integration [13], up to tailoring the collaboration or the development process of the socio-technical system itself [6][17]. Further, tailorability is directly related to Non-Functional Requirements (NFRs) in the case of integration, or indirectly such as in the case of customisation (i.e. usability concern), or extension (i.e. architectural concern). In the following discussion, we address how most prominent mobile applications are supporting tailorability in their provided location-related functionality.

Foursquare[8] is a social network designed for mobile devices supporting GPS. It allows registered users to connect with other people and track their location. ”Check ins” at venues are rewarded with user points etc. and can also be posted on Facebook[9] and Twitter[10]. Foursquare rewards active users by granting them ranks depending on their number of check-ins at venues. With increasing rank, the user’s authorizations increase as well. Examples for such additional authorizations are editting venue info, adjusting venue position (latitude and logitude) and adding categories to venues.

Google Places[11] integrates with Google Maps as well as Google+ and offers mainly information about businesses of all kinds. Business owners can create an entry for their location free. For mobile workers, or due to privacy concerns of people working from home, it is also possible to specify not an exact location, but an area of operation instead.

Other services, like e.g. Qype[12] and Yelp[13] focus on the rating and review of locations like restaurants, shops etc. Location information can be maintained by business owners themselves or by site visitors. In the second case, moderator approval is necessary. The service SCVNGR[14] is a location-based game platform. In order to get points, users are encouraged not just to visit certain places, but also to perform challenges there, which are being created by other users. This can be e.g. to take a picture, or answer a riddle etc. Geoloqui[15] offers a framework to build location-based applications. It offers functionalities like tracking, or definition of areas triggering an event. It is also possible to search in their POI database or create and store own private POIs, messages etc. on their servers.

Related to security and privacy needs, Palen and Dourish [15] mention that some level of information disclosure is needed to sustain social engagement. However, most available socio-technical systems have either a server-centric architecture, or a user-centric/client-centric architecture. User-centric approaches are not sufficient and suitable for social settings since the exchange of information is the base of such settings. Server-centric approaches imply that the server is the central point of information exchange. Such approaches do not fully eliminate accidental or intentional risks and threats, which can arise through the analysis or reconstruction of personal information and interaction

[7] Proper matching between the task and technological support

[8] https://www.foursquare.com/
[9] http://www.facebook.com/
[10] http://twitter.com/
[11] http://www.google.com/places/
[12] http://www.qype.com/
[13] http://www.yelp.com/
[14] http://www.scvngr.com/
[15] https://geoloqi.com/

traces. The building of a user's fully-fledged profile remains possible at least through the judicial authorities which enforce, for example, service providers to allow dispute resolution means in order to recognise frauds.

PrimeLife continues the work that has been done in PRIME and tried to tackle substantial new privacy challenges such as protecting privacy in collaborative scenarios and virtual communities by providing a privacy policy language and mechanism to handle access control, and to process privacy policy and compare them with user privacy preferences [19]. Our approach differs from identity management systems such as PRIME and PrimeLife in that it does not hinge on an identity management system –even if it is considered as trustworthy.

Furthermore, service providers impose their privacy policies to overcome privacy-related issues. They enforce their business models and interests that are based on private information and social interaction disclosure. Even though, a service provider applies different privacy-enhancing mechanisms such as anonymization (i.e., through removing sensitive data like names, addresses, etc.); there is still space for improvements since anonymization does not realize privacy and the service user must trust the service provider. In Narayanan and Shmatikov [14], it was shown that users could be re-identified across high popular distinct social networks like Facebook, Flickr, MySpace or Twitter with an error rate of just 12%. An interesting approach unifying centralized as well as decentralized aspects can be found in [4]. Bourimi et al. [4] present a decentralized group-centric approach, which empowers the users to host their environmental system needed for collaborative settings, instead of hosting it on a central server. Thus, the end-users have the full control over their data. The communication between different groups will be ensured over the main platform of the system. The user who hosts a surrounding node can share data with other groups without losing the control over his data. Bourimi et al. [4] approach builds the starting point for parts in our requirements analysis.

In summary, to our best knowledge, none of the surveyed mobile applications and approaches supports the definition of end users' own spots, beyond predefined categories. Further, no application fulfils our two main requirements, the on-the-fly definition of communication and/or data servers, as well as a flexible identity management concept.

5 Approach

Our approach is a combination of two conceptual approaches as well as a corresponding user interface. This interface supports end-users in defining and managing their own POI for their shared workspaces by using their preferred distributed architecture (i.e. global client/server or group centric etc.), according to their privacy preference. In order to fulfil the identified requirements HLR1 and HLR2 our approach includes:

(1) With the aim of meeting the runtime end-user tailorability requirement (HLR1) further concrete requirements were identified. An example of such requirement could be the interoperability and a flexible identity management system. The advantage of a flexible identity management system lies in the fact that such a system acts in the background and supports the changing process of the different server-side components in order to use them with other identities, without losing the relationships to the context at the client level. Furthermore, the used server-side POI collaboration components (e.g. for awareness, communication or data sharing) have to seamlessly support this switching of identities originating from the same client. These further concrete requirements are addressed for a specific prototypic implementation in the following section.

(2) The system architecture used to support the social interaction allows the separation of different components to be used in a given social setting/context (HLR2). So the end-users are able to adapt the used server components by themselves (e.g., communication, awareness or POI data sharing) according to their privacy needs from the same application at runtime without restarting the system or client application. It will be carried out by enabling the user to choose the server components needed for the social interaction (e.g., by entering their URLs). Furthermore, the end-users have the possibility to use ad-hoc servers set up by trusted people.

The server is used to administrate the application settings for the generic POI environment. Now, we only support one ”application” per installation, but this will be enhanced in the future to support multiple applications in one server installation. The application administrator configures the application. The administrator can upload an applications image, set the description, imprint for the application, and edit additional information. In our case, the customisation is specialized for iPhone application, but it can be more generalized for other mobile devices as well. The administrator has the opportunity to customize the generic POI objects, so that the application supports several types of POI. Therefore, the administrator needs to generate those customs POI by adding a POI name, an icon and (in the future) actions that are associated with that POI. In addition, the custom POI can have additional fields attached to them, which will be connected to the specific POI. For displaying purposes on the mobile devices, HTML templates can be associated to the POI. Each ”application” supports the use of groups, which can be shared by users. These groups can be public or private and allow POI to be attached to these groups. Each group can have multiple users attached to them. Users are able to create groups and POI through REST services, which are used by the mobile application.

Scenario 1 - Create new POI
A new POI is created by tabbing on the plus button in the upper left corner. ([create poi]) It is positioned in the

middle of the screen but can be moved via drag and drop afterwards. The POI is created with default values for name, description and type. This enables the user to quickly add a POI with minimum effort. If the user tabs on the POI, a callout-bubble appears which displays the name and type of the POI. The POI can be customized by clicking on the blue button on the right of the bubble.

Scenario 2 - Aggregate POI
To aggregate POI, the user first has to create the POI he wants to aggregate (see scenario one) and open the POI customisation window for the super-POI. The POI customisation window enables the user to set a name and description for the POI, change its type, share the POI with other users and add sub-POI to the current POI. If the user tabs on the ”Manage Sub-POI” button he can choose multiple POI from a list.

Scenario 3 - Filter POI
The user can choose which types of POI he wants to see on the Map by tabbing on the ”Filter icons” button in the map view [create poi]. In a list of POI types, he then can switch the desired types on or off.

5.1 Generic POI data model

In order to formalize the POI representation, we have defined the meta-model depicted in Fig. 4 using the Ecore[16] dialect of MOF[17]. It shows the abstract syntax of the language that is used by the server to model the POI information.

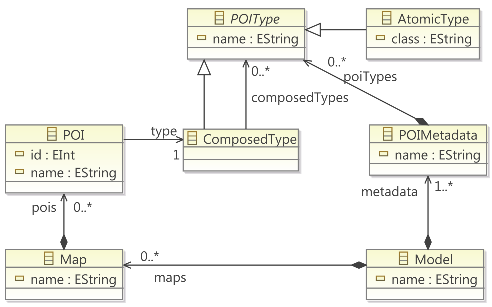
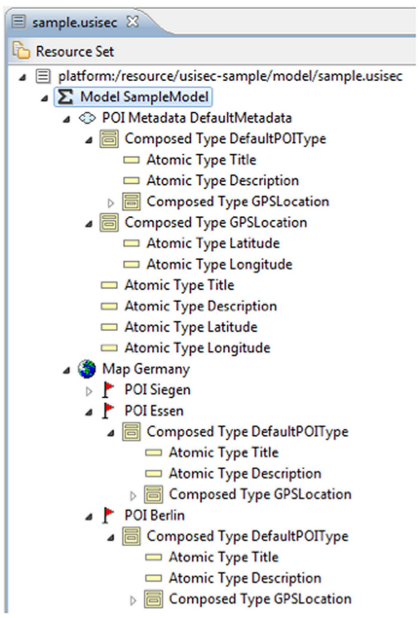


Fig. 4 Abstract syntax of server-side representation for POI in Ecore dialect

The **Model** is divided into two main aspects: the **Maps** and **POIMetadata**. While **Maps** represent the location context of **POI**, the **POIMetadata** defines the metadata related to the information that **POI** represent. A **POIMetadata** is defined in terms of **POITypes** that represent the metadata of information that is related to a **POI**. The **POIType** is defined as a Composite [7] to



sample.usisec

Resource Set

- platform:/resource/usisec-sample/model/sample.usisec
  - Model SampleModel
    - POI Metadata DefaultMetadata
      - Composed Type DefaultPOIType
        - Atomic Type Title
        - Atomic Type Description
        - Composed Type GPSLocation
      - Composed Type GPSLocation
        - Atomic Type Latitude
        - Atomic Type Longitude
      - Atomic Type Title
      - Atomic Type Description
      - Atomic Type Latitude
      - Atomic Type Longitude
    - Map Germany
      - POI Siegen
      - POI Essen
        - Composed Type DefaultPOIType
          - Atomic Type Title
          - Atomic Type Description
          - Composed Type GPSLocation
      - POI Berlin
        - Composed Type DefaultPOIType
          - Atomic Type Title
          - Atomic Type Description
          - Composed Type GPSLocation

Fig. 5 Meta model defining DefaultMetadata and POIMetadata

support: the definition of complex metadata and the reuse of metadata definition.

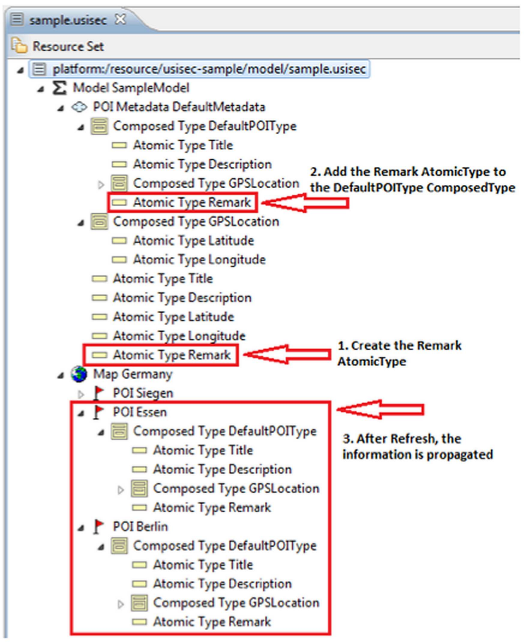
Figure 5 depicts a model that defines: (a) the *DefaultMetadata* **POIMetadata** for the Germany **Map**. Therefore, the *DefaultMetadata* **POIMetadata** defines the *DefaultPOIType* **ComposedType** that is subsequently defined by the *Title* and the *Description* **AtomicTypes** backed by the ”java.lang.String” class defined by the class attribute. Besides, the *DefaultPOIType* defines the *GPSLocation* **ComposedType** defined by the *Longitude* and *Latitude* **AtomicTypes** backed by the ”java.lang.Double” class defined by the **class** attribute. As can be seen in Fig. 5, all **POITypes** are refined at the **POIMetadata** level to encourage reuse. Thus, the *Germany* **MAP** defines the *Siegen*, *Essen* and *Berlin* **POI** that manage the information defined by the *DefaultPOIType* that is contained by the *DefaultMetadata* **POIMetadata**.

One of the goals of the system is the propagation of new metadata through the POI definition. Figure 6 shows how the addition of the *Remark* metadata in the *DefaultPOIType* **ComposedType** is propagated through the *Siegen*, *Essen* and *Berlin* **POI**.

5.1.1 Enhancement of Privacy by Enabling End-User Tailorability of the Distributed Architecture (HLR2)

We decided to use a group centric architecture with two lightweight servers. This server can be easily installed on a home computer or other hardware from the users

[16] http://www.eclipse.org/modeling/emft/?project=ecoretools
[17] http://www.omg.org/spec/MOF/


sample.usisec
Resource Set
platform:/resource/usisec-sample/model/sample.usisec
Model SampleModel
POI Metadata DefaultMetadata
Composed Type DefaultPOIType
Atomic Type Title
Atomic Type Description
Composed Type GPSLocation
Atomic Type Remark
Composed Type GPSLocation
Atomic Type Latitude
Atomic Type Longitude
Atomic Type Title
Atomic Type Description
Atomic Type Latitude
Atomic Type Longitude
Atomic Type Remark
Map Germany
POI Siegen
POI Essen
Composed Type DefaultPOIType
Atomic Type Title
Atomic Type Description
Composed Type GPSLocation
Atomic Type Remark
POI Berlin
Composed Type DefaultPOIType
Atomic Type Title
Atomic Type Description
Composed Type GPSLocation
Atomic Type Remark

2. Add the Remark AtomicType to the DefaultPOIType ComposedType
1. Create the Remark AtomicType
3. After Refresh, the information is propagated

Fig. 6 Example for addition of Remark metadata in Default-POIType

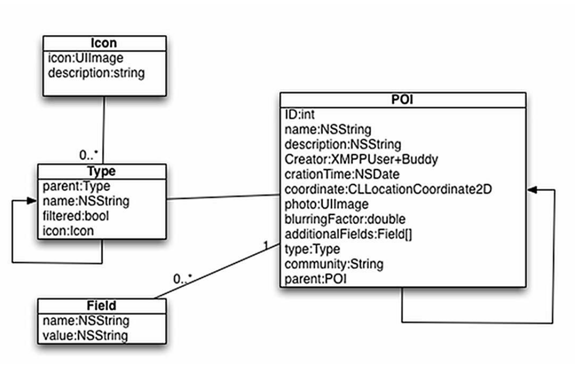


Fig. 7 Overall architecture of the prototype

themselves, if the user does not trust the owner of the public server and wants to have full control over his user data. Additionally, it allows the formation of sub-communities. The sub-communities can restrict access to their data so that only members of the community can access it. The two servers we used in iAngle, are the eJabbered Server for communication and location publishing, and a retrofitted CURE Server [12], mainly containing the database of watercourses and fishing spots. However, additionally involved in the registration process, to support unlinkabilty of communication data and user identities. Since the retrofitted CURE supports ubiquitously in form of decentralised group-centric servers, we developed many decentralised solutions. The iAngle client can be set to use an eJabberd server locally installed by the users themselves (members building a trust- worthy sub community) [3]. However, the central community AnglersBase is still being developed in PICOS,[BU3] and there is at the moment no possibility to share data at a more global level as described in [3]. Currently, the iAngle server is playing the role of the AnglersBase. Another important aspect in our approach is the fact, that no sensitive user information is stored on a server. The data like location information, email etc. are stored on the own mobile device and only sent to authorised contacts in an encrypted message. We might have a slightly increased communication traffic compared to other architectures, where the data is uploaded to a server, but the user possesses full control about his personal data all the time. The pseudonyms used for entering the iAngle server, where watercourses, and precise as well as blurred spots are stored, are very different from the eJabberd accounts. With this, the observability and linkability of the users are made difficult especially by seperating the communication as well as awareness functionality from the collaborative LBS scenarios.

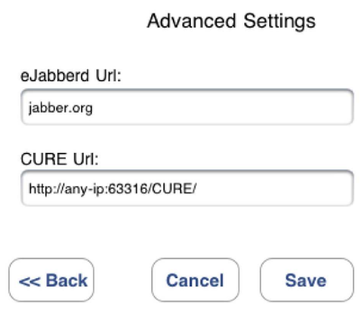

Advanced Settings
eJabberd Url: jabber.org
CURE Url: http://any-ip:63316/CURE/

Fig. 8 Tailoring the distributed architecture by setting the used server components

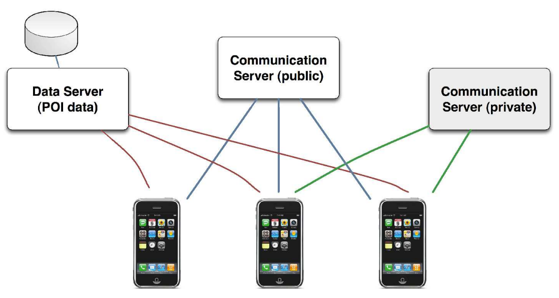


Fig. 9 Flexible and tailorable overall distributed architecture

By setting different values in the mobile App settings (see Figure 8) for the used servers one could become different distributed architecture allowing so for tailoring it to the respective end-users' privacy needs (meaningful constellations depicted in Figure 9).

6 Implementation

We implemented our approach for supporting a mobile Angling Community with privacy and collaboration needs related to location-based services. We built an iPhone based prototype based on our conceptual architecture introduced in [3], which allows for different levels of centralisation and decentralization, see Figure 9. Here we describe how the end-user can adjust (adapt) the distributed architecture in order to reach his/her privacy needs related to communication and awareness on the one hand, as well as shared data expectations on the other hand. We achieve this goal by enabling the user to configure the application architecture individually on the client-side (see Figure 8). Depending on the user's configuration, there are several architectures possible. In the following, we will introduce three possibilities.

Figure 10(a) shows a client-server architecture with the server being central for all users. Communication, awareness and data sharing is global in this configuration and therefore privacy is a global concern. The typical user of this configuration, has no or just a minor, interest in privacy. This model is also widely used by today's social networks. The next possibility is presented in Figure 10(b). In this case, all users share their data globally while communication and awareness aspects are handled group-centric. This model allows sharing contents publicly while respecting the user's privacy. Finally, the third configuration (see Fig. 10(c)) is very group-centric. Data is shared on the group-level and furthermore communication and awareness are handled on the group-level. The usage of XMPP with the help of the eJabberd server as a communication and awareness server granted the interoperability while CURE is used as a shared artefacts server. The end-users machine can host both the ubiquitous CURE (including an eJabberd server) [4] and as well as further ad-hoc setup eJabberd servers (ca. 12 MB). If the client switches for the first time to an eJabberd instance, the client application will create temporary identities and uses them. In the case of CURE, we enforce the user to enter the correct credentials through a separate UI. Since we only retrofitted the CURE implementation, the provided implementation details correspond to those described in [4].

6.1 Server implementation

The server part of the application is in charge of managing POIs information. The server manipulates two different types of information related to POIs

–The POI meta-information
–The POI information

While the POI meta-information defines the information to be stored according to the type of POI to be described, the POI information describes the POI itself according to the description provided by the POI meta-information. However, both types of information are processed in the same way through three different layers:

1. The Apache Tomcat Server
2. The Eclipse runtime environment
3. The MySQL database management system

The Figure11 depicts how the information is processed among these layers.

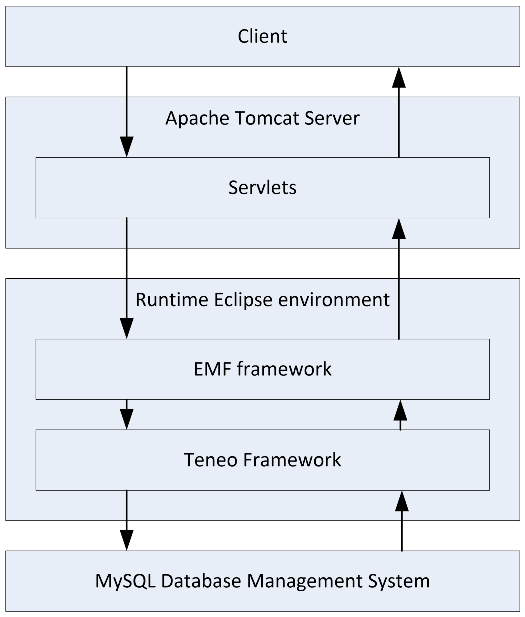
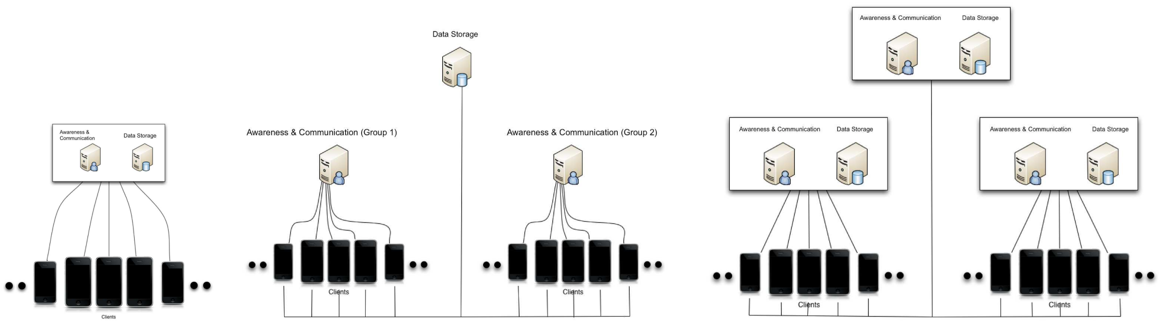


Fig. 11 Server software architecture

6.2 Apache Tomcat Server

The first layer is in charge of providing clients with information through the network using the HTTP protocol. To carry out this task, we employed an Apache Tomcat Server [18] which implements the Java Servlet technology[19]. Each operation performed by clients on the system is processed by a Servlet, which implements a REST web service. This Servlet is in charge of processing parameters and call the operations to be performed by the second layer in order to process the information. To carry

[18] http://tomcat.apache.org/
[19] http://www.oracle.com/technetwork/java/index-jsp-135475.html



Awareness & Communication
Data Storage
Clients
Awareness & Communication (Group 1)
Awareness & Communication (Group 2)

(a) Server-centric architecture (b) Architecture for group-centric communication (c) Group-centric architecture (awareness, communication, and shared data)

Fig. 10 The three different architecture approaches [3]

out this task, the Apache Tomcat Server is initialized according to the following code that is part of the ContextListener class:

```
package usisec.listeners;
...
import usisec.persistence.UsisecDBStart;

public class ContextListener implements
  ServletContextAttributeListener,
  ServletContextListener {
  ...
  @Override
  public void contextInitialized
    (ServletContextEvent arg0){
    UsisecDBStart instance = new UsisecDBStart();
    instance.doStartDB();
    arg0.getServletContext().setAttribute("instance",
    instance);
  }
  ...
}
```

Once, the application context is initialized with the instance of the Eclipse runtime environment (UsisecDBStart), the Servlet maps the operation to be performed and processes HTTP parameters. Then, it calls the operation to be performed on the Eclipse runtime environment (i.e. the **addPOI** operation) The following code shows how this process is carried out.

```
package usisec.servlets;
...
public class AddPoiData extends HttpServlet {
  ...
  @Override
  protected void doGet(HttpServletRequest req,
    HttpServletResponse resp) throws ServletException,
      IOException {
    // Idem doPost
  }
  @Override
  protected void doPost(HttpServletRequest req,
    HttpServletResponse resp) throws ServletException,
      IOException {
    UsisecDBStart instance = (UsisecDBStart)
      getServletContext().getAttribute("instance");
    ...
      // Get HttpServletResponse writer
      ...
      // Process parameters
      java.util.Map<EAttribute, Object> values
        = instance.getPOIMetadata();
      for (EAttribute attr : values.keySet()) {
        values.put(attr, req.getParameter(attr.getName()));
      }
      // Calls Eclipse runtime operation
      instance.addPOI(values);
      // Generate REST response by asking the instance
      ...
      // Close HttpServletResponse writer
      ...
  }
}
```

6.3 Eclipse Runtime Environment

The second layer is based on an Eclipse [20] runtime environment which hosts the model instances to be manipulated by the Servlets hosted in the Apache Tomcat Server. Model instances represent instances of the meta-model meta-classes represented in Figure 4. To deal with the creation and modification of these model instances, we have used the Eclipse Modeling Framework (EMF) [21] which runs in the Eclipse runtime environment. By default the persistence of model instances are stored in XML using the XMI (XML Metadata Interchange) format [22]. This way of storing information may be useful for single threaded applications; however, it is not the right choice in multi-threaded environment as the web environment. Therefore, we have used the Teneo persistence framework to support model storage, using database management systems. Teneo [23]

is a database persistence solution for EMF using Hibernate [24] or EclipseLink [25]. It supports automatic creation of EMF to Relational Mappings. EMF Objects can be stored and retrieved using advanced queries (HQL or EJB-QL).

20 http://www.eclipse.org/
21 http://www.eclipse.org/emf
22 http://www.omg.org/spec/XMI/
23 http://www.eclipse.org/modeling/emft/?project=teneo#teneo
24 http://www.hibernate.org/
25 http://www.eclipse.org/eclipselink/

Thus, the following code shows how the Eclipse Runtime Environment processes the request using the EMF.

```
package usisec.persistence;
...
public class UsisecDBStart {
  ...
// Initialization
  ...
  public void addPOI
    (java.util.Map<EAttribute, Object> values) {
   Session session = sessionFactory.openSession();
   Transaction tx = session.getTransaction();
   // Starts a transaction, create a library
   and make it persistent
   tx.begin();
   Query qry = session.createQuery("from Model");
   List<?> list = qry.list();
   // Retrieves the model (root)
   Model model = (Model) list.get(0);
   // Creates a POI
   POI poi = UsisecFactory.eINSTANCE.createPOI();
   // Retrieves parameters
   for (Map.Entry<EAttribute, Object> entry :
      values.entrySet())
     poi.eSet(entry.getKey(), entry.getValue());
   // Saves POI information
   session.save(poi);
   // Adds a new POI to the model
   model.getPois().add(poi);
   // at commit the objects will be present in the
 //database
   tx.commit();
   // and close of, this should actually be done
   //in a finally block
   session.close();
  }
}
```

Finally, the third layer is implemented by the MySQL [26] Relational Database Management System.

6.4 Advantages of this approach

As result of this implementation, we have achieved a multi-platform approach for lightweight clients.

One advantage of this approach, is the runtime updating mechanism due to the meta-modelling conception of the system. In a traditional approach you have to modify domain classes in order to add new meta-information to the system, for instance, following the scenario described on Section 5.1 (Figures 5 and 6) you have to manually add the "remark" attribute to the POI type, adjust database table fields accordingly, re-compile and restart the system in order to reload classes.

However, the use of the EMF jointly with the Teneo framework allows the modification of the meta-information in the same way we modify instance information achieving a runtime reflection system.

Thus, the type of a POI (POIType) is linked to the POI instance, which is treated as simple information itself; consequently, the variation of the POIType modifies the POI itself. Therefore, no recompilation or restart is needed when meta-information is modified.

[26] http://www.mysql.com/

As we have mentioned before, the use of the EMF jointly with the Teneo framework allows transactional operations on models, which provides the system with reliability and efficiency when dealing with information storage.

The multi-layer approach also allows developers to change the database management system easily because Teneo abstracts the persistence layer from the EMF framework. Besides, this approach allows developers to change the Servlet implementation because the EMF runs independently from the web container being used.

7 Conclusions and Future Work

This work motivated and presented a generic framework for end-user tailorability of the UI as well as server-side for POI functionality by considering privacy aspects. We identified the need for supporting end-user tailorability based on performed analysis of various use cases related to collaborative location-based scenarios in mobile settings from different projects. The two high-level requirements identified in this paper are related to supporting generic POI and end-user privacy needs by allowing for tailorbility. We demonstrated the feasibility of the generic framework for collaborative mobile applications and services that support privacy-respecting location-based scenarios by means of an iOS based prototype being used in the iAngle and iFishWatcher projects. The iOS based mobile App communicates with a generic meta model at the server-side, supporting the creation of POI at-runtime (here by using Eclipse's metamodeling framework). Our prototype allows different communities to define their own points of interests in a generic manner (at-runtime) by simultaneously supporting group collaboration functionality (e.g. communication, awareness, etc.). Thereby, the distributed architecture can also be tailored according to privacy needs.

Our work goes beyond related work. Future work will focus on improving the end users' tailorability capabilities; so that lay users are empowered to easy generate their specific community Apps by using our generic framework. For instance, the same framework can be re-used by other communities (not just our Angling Community) and its UI can be tailored to use other pictures and icons by keeping the location-based and collaborative functionality unchanged. This work began since some weeks to perform ethnographic lab evaluations for detecting crucial usage points (i.e., critical points for user experience). One of the main point is to provide support for good tailorability user experience in the next versions.

Acknowledgement

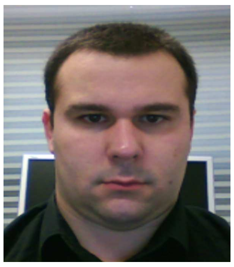
We acknowledge support by the Deutsche Forschungsgemeinschaft (DFG) under grant KE 1213/3-1. Further support was provided by the EU FP7 project digital.me, funded by the EC(FP7/2007 2013) under grant no. 257787.

References

[1] Liam J. Bannon. Customization and tailoring of software systems:thinking about the context of tinkering and tailoring. In *Customizing software systems*, 4–8 (1992).
[2] M. Bourimi, B. Ueberschaer, E. Ganglbauer, D. Kesdogan, T. Barth, J. Dax, and M. Heupel. Building usable and privacy-preserving mobile collaborative applications for real-life communities: A case study based report. In *Information Society (i-Society), 2010 International Conference on*, 435 –442 (2010).
[3] M. Bourimi, J. Ossowski, Dhiah el Diehn I. Abou-Tair, S. Berlik, and D. Abu-Saymeh. Towards Usable Client-Centric Privacy Advisory for Mobile Collaborative Applications based on BDDs. In *4th IFIP International Conference on New Technologies, Mobility and Security (NTMS)*, Paris, France, 7-10 (2011).
[4] Mohamed Bourimi, Falk Kühnel, Jörg M. Haake, Dhiah el Diehn I. Abou-Tair, and Dogan Kesdogan. Tailoring collaboration according privacy needs in real-identity collaborative systems. In *CRIWG*, 110–125 (2009).
[5] M. Dedual, O. Sague Pla, R. Arlinghaus, A. Clarke, K. Ferter, P. Geertz Hansen, D. Gerdeaux, F. Hames, S. J. Kennelly, A. R. Kleiven, A. Meraner, and B. Ueberschr. Communication between scientists, fishery managers and recreational fishers: lessons learned from a comparative analysis of international case studies. *Fisheries Management and Ecology*, 20 (2-3), 234–246 (2013).
[6] Alejandro Fernández, Jörg M. Haake, and Adele Goldberg. Tailoring group work. In *CRIWG*, 232–244 (2002).
[7] Erich Gamma, Richard Helm, Ralph Johnson, and John Vlissides. *Design Patterns - Elements of Reusable Object-Oriented Software*. Addison-Wesley Longman, (1995).
[8] Chandra Prasad Giri, Surendra Shrestha, Timotthy W. Foresman, and Ashbindu Singh. Global biodiversity data and information, (2009).
[9] Austin Henderson. Tailoring mechanisms in three research technologies. In *Proceedings of Group '97*, (1997).
[10] Hal Hodson. Smartphones make identifying endangered animals easy. *New Scientist*, (2013).
[11] Jason I. Hong and James A. Landay. An architecture for privacy-sensitive ubiquitous computing. In *MobiSys '04: Proceedings of the 2nd international conference on Mobile systems, applications, and services*, New York, NY, USA, 177–189, (2004).
[12] Stephan Lukosch and Mohamed Bourimi. Towards an enhanced adaptability and usability of web-based collaborative systems. *International Journal of Cooperative Information Systems, Special Issue on 'Design, Implementation of Groupware*, 467–494 (2008).
[13] Anders Mørch. Three levels of end-user tailoring: customization, integration, and extension. *MIT Press*, 51–76 (1997).
[14] Arvind Narayanan and Vitaly Shmatikov. De-anonymizing social networks. 173–187 (2009).
[15] Leysia Palen and Paul Dourish. Unpacking "privacy" for a networked world. In *CHI '03: Proceedings of the SIGCHI conference on Human factors in computing systems*, New York, NY, USA, ACM Press, 129–136 (2003).
[16] PICOS Consortium. D6.2b - Community Application Prototype, (2010).
[17] Till Schümmer. *A Pattern Approach for End-User Centered Groupware Development*. Schriften zu Kooperations- und Mediensystemen - Band 3. JOSEF EUL VERLAG GmbH, Lohmar - Köln, (2005).
[18] Robert Slagter. *Dynamic groupware services, modular design of tailorable groupware*. PhD thesis, University of Twente, (2004).
[19] Slim Trabelsi, Gregory Neven, and Dave Raggett. Privacy and Identity Management in Europe for Life: Report on design and implementation. Technical report, PrimeLife Consortium, (2011).

Dhiah el Diehn I. Abou-Tair is an assistant professor at the German-Jordanian University. He received his PhD from the group of Database and Software Engineering at the University of Siegen, Germany. During his PhD, Dr. Abou-Tair conducted research about the adoption of privacy laws and regulations in information systems through an ontology-based approach. He has wide expertise in the fields of domain analysis, ontology development, database modelling, integration of heterogeneous software systems and development of web based information systems. Dr. Abou-Tair was a post doctorate researcher at the Chair of IT-Security at the University of Siegen, and has been involved with a number of EU and German-funded research projects.

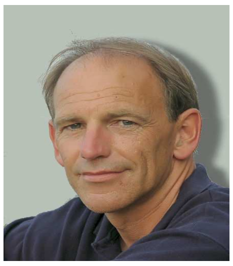
Mohamed Bourimi graduated from the University of Dortmund in 2002 and holds a degree in computer science Diplom-Informatiker with distinction). Mohamed is working now as a research assistant at the IT Security chair at University of Siegen since 2009. He contributed as developer, consultant as well as technical project leader to various German and EU research and industrial projects. He owns more than 30 international scientific publications and is certified in Scrum, ITIL v3, and in IBM Enterprise Technologies and Mainframes. Currently Mohamed is mainly contributing to the EU FP7 digital.me project as leader of WP4 concerned with the development of digital.me Trust, Privacy and Security Infrastructure.



Ricardo Tesoriero is professor at Computing Systems Department, University of Castilla La Mancha (UCLM), Albacete, Spain. He got a degree in computer science in 2005 at the National University of La Plata, Buenos Aires, Argentina, a master degree in Advanced Information Technologies in 2008 at the UCLM and a PhD in computer science in 2009 at the UCLM too. He is member of the Interactive Systems Everywhere Research Group of the Albacete Research Institute of Informatics. His teaching and research areas are Software Engineering and Human-Computer Interaction (HCI). He is co-author of more than 50 publications in journals, book chapters and international congress proceedings. His research interests are model-driven architectures, HCI and context-aware computing.

Marcel Heupel graduated from the University of Siegen with a degree in information systems (Dipl. Wirtsch. Inf.), with main focus on anonymity support at application level and its usability. Currently he is a PhD student at the IT Security Chair at Siegen. After his graduation he contributed to more than 10 publications until now, being first in author of four of them. Before he graduated he has already German national and international publications related to these anonymity topics as co-author. He is currently contributing in the EU FP7 funded project digital.me.

Dogan Kesdogan holds the Chair of IT-Security at the Universität Regensburg. His primary fields of research are security and privacy with the goal to provide a theoretical background for the development, implementation and evaluation of security and privacy-enhancing protocols. He is a graduate of the Aachen University of Technology where he has also received his doctoral degree and habilitation in computer science. He has held faculty and industrial positions at University of Siegen, NTNU Norway, RWTH Aachen, VU University Amsterdam, o.tel.o communications GmbH and IBM Thomas J. Watson Research Center.



Bernd Ueberschär is a marine biologist with the focus on sustainable management of fish resources, aquaculture, online information systems and the human dimensions of recreational fisheries. He was a research partner in the PICOS-Project and developed new privacy concepts for angling communities, assisted in the translation of these concepts into mobile online communities for recreational angler and supervised a series of field trials where related applications for smartphones were tested.