Abstract
This deliverable describes the patterns of interaction for the Internet of the Future, gives an overview of challenging usage scenarios and describes the two main application scenarios: eHealth and SalesEverywhere.

Deliverable details
Deliverable version: v1.0
Date of delivery: 30.09.2011
Editors: Jan Wolff, Siemens AG
List of Contributors:
Bastian Braun, Jorge Cuellar, Philippe De Ryck, Lieven Desmet, Martin Johns, Michael Kirchner, Peng Liu, Frank Piessens, Joachim Possega, Jan Stijohann, Walter Tighzert, Monica Verma, Jan Wolff

Project details
Start date: October 01, 2010
Duration: 36 months
Project Coordinator: Martin Johns
Partners: SAP, Siemens, CHALMERS, KUL, UNI PASSAU
## Contents

1. **Introduction**  
   - 1.1 Overview  
   - 1.2 Interaction Patterns and Security Properties  

2. **Interaction Patterns**  
   - 2.1 Definition and Terminology  
   - 2.2 Desired Properties  

3. **Pattern Catalog**  
   - 3.1 P01 Interaction with a Web Page  
   - 3.2 P02 Client-side Third Party Content Integration  
   - 3.3 P03 Client-Side Third Party Library Integration  
   - 3.4 P04 Third Party User Interface Integration  
   - 3.5 P05 Resource Reference Sharing  
   - 3.6 P06 Third Party User Tracking  
   - 3.7 P07 Geolocation Sharing  
   - 3.8 P08 Capability-based Authorization Delegation  
   - 3.9 P09 Distributed Workflow  
   - 3.10 P10 Out-of-band distributed service  

4. **Usage Scenarios**  
   - 4.1 S01 Google Maps  
   - 4.2 S02 Dangerous Goods  
   - 4.3 S03 User Tracking  
   - 4.4 S04 Advertisements  
   - 4.5 S05 Tax/Loan Calculator  
   - 4.6 S06 Third Party Libraries  
   - 4.7 S07 Interactive Avatar  
   - 4.8 S08 Third-party authentication  
   - 4.9 S09 Facebook Login  
   - 4.10 S10 Third-Party Payment  
   - 4.11 S11 Facebook Application  
   - 4.12 S12 Google Widget  
   - 4.13 S13 Social E-commerce  
   - 4.14 S14 Contact Swap  
   - 4.15 S15 Holiday Pictures  
   - 4.16 S16 Mobile Apps for Enhanced Security  
   - 4.17 S17 Delegation of Privilege  
   - 4.18 S18 Cross-domain Interaction  
   - 4.19 S19 SMS Portal  

---

FP7-ICT-2009-5  
Project No. 256964

![WebSand](https://www.websand.eu)
5 Use Case: e-Health 50
  5.1 e-Health Application Domain ............................. 50
  5.2 System Architecture ........................................ 52
  5.3 Workflows ..................................................... 55

6 Use Case: Sales Everywhere 59
  6.1 Application Domain ........................................... 59
  6.2 System Overview ............................................... 60
  6.3 System Architecture .......................................... 60
  6.4 Workflows ..................................................... 62

7 Conclusion 66
Executive Summary

This document is the first version of deliverable 1.2, due in project month M12. In the description of work, deliverable D1.2 is described on page 37. In the second version, the catalog of interaction patterns will be extended when new patterns emerge.

This deliverable centers itself mainly around Interaction Patterns and related aspects viz. desired properties, advanced characteristics, purpose, interaction work flow, security challenges, etc. These Interaction Patterns have been derived based on the challenging scenarios and use-cases gathered for the Internet of the Future applications.

In the following, the main parts of this deliverable are described and mapped to the requirements of the description of work. We start with an “Introduction” section 1 which describes briefly the Interaction Patterns. The introduction further talks about the uses cases, that have been described in details in this deliverable, and the co-relationational aspect of Interaction Patterns with frequently observed scenarios and use cases.

In the next section “Interaction Patterns” 2, we start with defining the basic terminology to understand what interaction patterns really are.

Section 3 delineates a catalog of interaction patterns for the Internet of the Future, which have been carefully considered, derived and categorized to cover and map to the the interaction scenarios described in Section 4 and the use cases described in Section 5 and Section 6. These Interaction Patterns help understand and serve as a groundwork for the plausible cross-domain interaction requirements, cross-domain trust boundaries and considerations, and technical and legal security requirements. Each of the Interaction Pattern is delineated with abstract overview, generic description, technical description, diagrammatic interaction work flow and specific examples.

Section 4 contains a set of challenging and innovative usage scenarios of the Internet of the Future. These scenarios have been selected by all partners as a representative set from the perspective of the main problem fields of mash-up security, which are addressed by the technical work packages: WP2 Secure Web Interaction, WP3 Information-Flow Control, and WP4 Secure Composition.

Section 5 contains the description of the use case WebHealth in the application domain e-Health.

Section 6 contains the description of the use case Sales Everywhere in the application domain Sales and Software-as-a-Service providing.
1 Introduction

This deliverable describes the patterns of interaction for the Internet of the Future and the two main application scenarios, that have been selected by the industrial partners SAP and Siemens. The interaction patterns are short, generic use cases which represent abstract parts of innovative and challenging usage scenarios in the scope of the WebSand project. Where possible those abstract interaction patterns are mapped to the specific scenarios which are described in deliverables D2.1, D3.1, and D4.1 and to the specific use cases and workflows of the main application scenarios that are also described in this document. In the context of mashups, the interaction patterns reflect the focus of the WebSand project on secure web interaction, information-flow control, and secure composition.

The main use cases eHealth and Sales Everywhere will be used in WP5 to integrate and evaluate the WebSand application security framework. Here, they are described on a high level from the perspective of the application domain to provide a basis for the analysis of legal aspects in WP1.3. Aspects that represent challenges for the technical work packages or that relate to the interaction patterns of the future internet are highlighted.
2 Interaction Patterns

This section describes commonly observed patterns of interaction which represent parts of innovative and challenging usage scenarios of Future Internet applications. These patterns have been carefully determined and categorized, with due consideration, based on

- the challenging usage scenarios gathered by the project partners which reflect the focus of the WebSand project on secure web interaction, information-flow control, and secure composition, and
- the aspects of the use cases related to challenging technical and legal security requirements.

Wherever possible those abstract interaction patterns are mapped to the specific scenarios that are described in Section 4, and to the specific use cases and work flows of the main application scenarios that are described in Section 5 and Section 6 of this document.

2.1 Definition and Terminology

Before diving into the various Interaction Patterns or IAPs, we introduce the related terminologies. In the websand WP1 context, we regard an Interaction Pattern (IAP) as

- a structured, generic representation of a frequently found interaction between two or more multi-party components and users. The interaction being either
  
  a short self-contained step being part of a more complex interaction sequence

  or a simple work flow being a building block of more complex work flows.

The overall goal of introducing and collecting IAPs is to review common mash up interactions and eventually identify possible security shortcomings. Due to the latter goal, an IAP serves as a structured description of a software engineering research task, comprising implicit functional and explicit security requirements.

A core part of an IAP representation is a model, which outlines the essential aspects of the interaction. It can be seen as a software design artifact illustrated by one or more UML diagrams. Similar to a design pattern, this
model is only a template whose *instantiations* are concrete implementations of the model which satisfy the functional requirements but not necessarily all security requirements. For reading convenience, we will sometimes do not distinguish between the IAP, i.e. the complete representation, and the belonging model.

### 2.2 Desired Properties

Many desired properties of IAPs are similar to those of (design) patterns used in software engineering:

- The representation has a **unified structure**
- but leaves flexibility to **cover different levels of abstraction**.
- IAPs are **generic** and can be **instantiated**.
- IAPs are **composable** to form more complex constructs.
- IAPs supports the **transition from problem space to solution space**:
  - A **common language** for problem (IAP) and solution (instantiation) is established.
  - An IAP introduces technical elements which can be re-used for the design of an instantiation.
- IAPs describe the essence of popular (web mash-up or web multi-party) interactions, i.e. their **instantiations are frequently found** in today’s web interactions.

### Advanced Characteristics

IAPs are supposed to be composable and have hierarchical relations. This means that simple basic and frequently occurring interaction steps may form basic ingredients for more complex IAPs. Examples for such **Base IAPs** are Interaction with a web site and Third party client-side content integration described at the beginning of the pattern catalog. This building block concept allows an efficient IAP representation, especially regarding the model and security requirements, as well as the creation of structured catalog of IAPs.
Degree of abstraction

A good IAP (description) has an appropriate degree of abstraction. This concerns the IAP model, i.e. the belonging diagrams (e.g. context diagrams, UML use case diagrams, UML sequence diagrams, ...) but also the textual descriptions. Possible are for example:

- Domain context language, e.g.
  “The user visits a travel web shop which has added a weather forecast service from weather.com to his website. He checks the weather forecast for Africa. He then adds a Safari-Trip to his shopping cart and confirms his order by sending his credit card information.”

- Neutral but not technical language, e.g.
  “The user visits a web site containing several 3rd party services. He sequentially uses some of these 3rd party services and finally sends a request to the web server based on the prior service interactions.”

- Technical language, e.g.
  “The client browser request a mash-up integrating website and the contained JavaScript code loads the widget code from all integrated 3rd party services. He then triggers a service via an on-mouse-click JavaScript event. The result of the first service request is stored in a cookie, which ...”

In general, a good IAP description is not too concrete, too technical or too specialized. Furthermore, a very domain specific description hinders a reuse in other situations and contexts. On the other hand, it is also not too abstract, too general, too generic or too non-precise because otherwise a solution would have no directly applicable benefit or would be not helpful at all. Moreover, not all possible appropriate solutions may be taken into consideration if the context is not specific enough. Consequently, a good IAP restricts the problem space as much as possible and the solution space as little as necessary.
### 3 Pattern Catalog

In the following, the interaction patterns are described following the definition and terminology defined in the previous section.

The following table lists the identified patterns, their inter-relations and relations to usage scenarios in Section 4.

<table>
<thead>
<tr>
<th>ID</th>
<th>Interaction Pattern</th>
<th>Related Patterns</th>
<th>Related Scenarios</th>
</tr>
</thead>
<tbody>
<tr>
<td>P01</td>
<td>Interaction with a Web Page</td>
<td>P02 – P10</td>
<td>S01 – S19</td>
</tr>
<tr>
<td>P02</td>
<td>Client-side Third Party Content Integration</td>
<td>P01, P03, P04, P07, P09</td>
<td>S01, S02, S04, S05, S06, S07, S09, S12, S13, S15, S18</td>
</tr>
<tr>
<td>P03</td>
<td>Client-Side Third Party Library Integration</td>
<td>P01, P02, P06</td>
<td>S06</td>
</tr>
<tr>
<td>P04</td>
<td>Third Party User Interface Integration</td>
<td>P01, P02, P05, P07, P08, P09</td>
<td>S01, S02, S04, S07, S12, S15</td>
</tr>
<tr>
<td>P05</td>
<td>Resource Reference Sharing</td>
<td>P01, P04</td>
<td>S04</td>
</tr>
<tr>
<td>P06</td>
<td>Third Party User Tracking</td>
<td>P01, P03</td>
<td>S03, S04</td>
</tr>
<tr>
<td>P07</td>
<td>Geolocation Sharing</td>
<td>P01, P02, P04</td>
<td>S01, S02, S05, S15</td>
</tr>
<tr>
<td>P08</td>
<td>Capability-based Authorization Delegation</td>
<td>P01, P04</td>
<td>S11, S17</td>
</tr>
<tr>
<td>P09</td>
<td>Distributed Workflow</td>
<td>P01, P02, P04</td>
<td>S08, S09, S10, S17</td>
</tr>
<tr>
<td>P10</td>
<td>Out-of-band distributed service</td>
<td>P01</td>
<td>S16</td>
</tr>
</tbody>
</table>

Table 1: Interaction Patterns and their relations.
3.1 P01 Interaction with a Web Page

Overview

A user actively navigates to a web page and interacts with it by consuming content or sharing information with the page.

Actors and important System Components

- **User**: The person using a web browser to access a web page. The user is interested in interacting with the web page by retrieving information from or sharing information with the page.

- **Web Browser**: The client-side Agent that acts on the behalf of the user and sends the requests on the behalf of the user to the web server. The response (information/multi-party content) sent by the web server is rendered and presented to the user by the web browser.

- **Web server**: The web server represents the content provider that delivers the web page.

- **Web page**: The web page is served by the primary web server and rendered by the browser of the user.
Interaction Outline

1. The user initiates a request via his browser to load a certain URL.

2. The web server returns the web page in form of HTML code and possibly other elements (images, scripts, ...)

3. The browser processes the data and renders it so that it eventually can be read by the user.
3.2 P02 Client-side Third Party Content Integration

Overview

A user visits a web page which consists of elements provided by the corresponding primary web server but also contains content from a third party web server. The user may or may not be aware of the existence of third party content.

Pattern extends: P01

Actors

- **User**: The person using a web browser to access the primary web page. The user may not have knowledge about which elements on the primary web page originate from a third party.

- **Primary web page**: The primary web page is served by the primary web server and rendered by the browser of the user. As a part, it contains third party content.

- **Primary web server**: The content provider that delivers the primary web page.

- **3rd party web server**: The 3rd party web server provides content that is integrated into the primary web page.
Interaction Outline

1.-3. See base pattern “Interaction with a Web Page”

4. As part of the processing and rendering of the primary web site, the browser notices that third party content is to be integrated and sends according requests to the third party web server.

5. The third party web server returns the requested content which is then integrated by the browser into the primary webpage.
3.3 P03 Client-Side Third Party Library Integration

Overview
A user visits a web page which leverages functionality provided by a third party in form of a code library. The functionality is rather basic and tightly integrated with the website, i.e. the user is not able to distinguish between primary web page and third party functionality, or, may not be aware of the integration of third party functionality at all.

Pattern extends: P02

Actors
- **User**: The person using a web browser to access the primary web page. The user may not have knowledge about which elements on the primary web page originate from a third party.
- **Primary web page**: The primary web page is served by the primary web server and rendered by the browser of the user. Not visible to the user, this page leverages a third party library.
- **Primary web server**: The content provider that delivers the primary web page.
- **3rd party web server**: The 3rd party web server provides the library that is used in the primary web page.
- **3rd party JavaScript Library**: A set of functions and data. The primary web page just includes a reference to the library, it is loaded dynamically when the browser processes the primary web page.

Examples
Several JavaScript libraries such as jQuery and Dojo exist that contain pre-made software constructs and functionality. A third-party provider offers its own standard API that is embedded into the primary website and acts as a holder for the library. The properties, functionality and features of the third-party library is controlled and governed by the provider itself. The library may be customized by the provider to suit the requirements of the specific user.
The widgets are mainly third-party JavaScript code that are embedded into the primary website and act as a holder for third-party content integrated
into the primary web page. The content to be displayed is governed by the widget property and its API.
3.4 P04 Third Party User Interface Integration

Overview

A user visits a web page which contains a third party user interface, i.e. the page integrates third party content which is clearly visible to the user and he is able to tell its origin.

Pattern extends: P02

Example Instantiations

Many third parties provide standard user interfaces for integrating their content into other web pages. Such widgets serve as read-only streams. Possible belonging content is available in the read-only view. However, the user interface may additionally facilitate certain client-side events/actions, e.g. relocation of the markers on Google Maps i.e. the widget API may provide certain user-customizable features, however in a restricted manner, the primary control still stays with the provider. Known example instantiations are:

1. Twitter Widget: displays user A’s status updates using Twitter’s Status Timeline API which is then embedded in the primary webpage.

   Definition: Display your most recent Twitter updates on any webpage[6]. The following JavaScript code is embedded in the primary webpage:

   ```html
   1 <script src="http://widgets.twimg.com/j/2/widget.js"></script>
   2 <script>
   3 new TWTR.Widget({
   4   version: 2,
   5   type: 'profile',
   6   rpp: 5,
   7   interval: 6000,
   8   width: 250,
   9   height: 300,
   10  theme: {
   11    shell: {
   12      background: '#333333',
   13      color: '#ffffff'
   14    },
   15    tweets: {
   16      background: '#000000',
   17      color: '#ffffff',
   18      links: '#99a0eb'
   19  }
   20  })
   ```
The `widget.js` calls twitter’s user_timeline api to return the most recent statuses either of the authenticating user or of another user by providing his `screen_name` or `user_id`[5]:

```
http://api.twitter.com/version/statuses/user_timeline.format
```

Return Format: json, xml, rss, atom
Cross-Domain Access: Dynamic scripting, refered to as JSONP in case of JSON output format

2. Google Widget

3. Weather Widget
3.5 P05 Resource Reference Sharing

Overview

Resource reference sharing refers to sharing the reference of a resource, for example a photo or text, with other users via integrated third party functionality.

Pattern extends: P04

This sharing feature is available in form of content elements that are embedded within the resource website. The user can then interact with these elements and thus share the reference, e.g. by posting the reference on a platform offered by the same third-party. This means, to successfully share a reference, the user must have an account or profile with the third party provider. The same may hold for users that want to see the shared reference. They also need to have access to that 3rd party platform, possibly in form of an account or profile.

Actors

- **User:** The end-user that, via a browser, accesses a primary web page and makes use of the resource reference sharing functionality.

- **Primary web server:** The primary web server represents the content provider that delivers the primary web page. The user shares a reference to a resource contained in that web page.

- **Primary web page:** The primary web page is served by the primary web server and rendered by the browser of the user. The primary web page contains third party content elements that implement the resource reference sharing functionality.

- **3rd party web server:** The 3rd party web server provides the content elements which are necessary for the resource reference sharing functionality. In addition, the reference to be shared is stored on a web page on his web server.

Interaction Outline

An overview of the user interaction flow can be described as follows:

1.-5. See base pattern “Client-side third Party Content Integration”
6. The user initiates a sharing request, e.g. by clicking a button.

7. The sharing request is sent to the 3rd party together with the reference to share.

8. Depending on the 3rd party platform, the user has to take further actions, e.g. log-in, to enable the publishing of the reference.

**Example Instantiations**

With more and more social networks emerging and various providers stepping into the social media market, the number of providers offering the feature to “share-bookmarks” is multiplying rapidly. Facebook, Google, Twitter, Digg, LinkedIn, and many more provide users the possibility of sharing bookmarks of various resources, they find interesting, across their social media profile. Facebook Like, Google +1, Digg share, Delicious bookmark, LinkedIn share, just to name a few, are available in form of HTML buttons that can be embedded via JavaScript code within the resource website. These social plugins/widgets, simply post/share the bookmark on user’s profile at the social media platform.
The above overview of the interaction can be described in details with focus on the technological aspects. Figure 1 describes the sequence flow of the interaction in details.

To allow the users to bookmark your resource on their social account, you simply embed the third-party JavaScript code (or the iFrame code as a variant in certain cases) for the social bookmark plugin into your website. On request for the resource, the user receives the desired website containing the resource and the embedded code for the bookmark plugins. An example of that would be a blog with embedded Facebook Like, Twitter Tweet and Google +1 social plugins. This embedded code has access to the webpage’s DOM-API and knows the url for the resource that is desired to be bookmarked. Assuming that the user is logged onto his social media account, clicking the plugin causes a request to be sent to the third-party social web server that adds the bookmark to the user’s account. For e.g. Facebook Like button adds the bookmark to user’s Facebook profile stating that the user “likes” it.

1. Twitter’s Tweet
   Technology: Both JavaScript and iFrame
   Supports both synchronous and asynchronous calls

2. Facebook’s Like
   Technology: Both JavaScript and iFrame

3. Google’s Recommend
   Technology: Only JavaScript

A particularly interesting case arises when a youtube video is “liked”. In that case, not only is the bookmark integrated but the Facebook server fetches the video from the primary web server and integrates (embeds) the video content into the user’s profile page.

Refer to Figure 2
3.6 P06 Third Party User Tracking

Overview

Third Party User Tracking refers to the integration of a third-party library to track and collect various information about a user’s web activity. In most cases, this happens without the user’s knowledge and consent.

Pattern extends: P03

User Tracking is a pragmatic solution in understanding user activity behavior and preferences. It is becoming an ever-growing source of revenue via customized advertisements displayed to users according to their preferences. Tracking of user’s behavior depends on various sources and mechanisms used in the process such as

1. IP Address of the user, which can be used for IP Geolocation
2. domain names (sub-domain names)
3. first-party cookies
4. keywords of the content of the website being tracked
5. search query keywords / results
6. cache of the page
7. DOM objects
8. HTTP Request: browser type, hostname, referer

**Example Instantiations**

A popular instantiation is Google Analytics. The interaction pattern for Web Analytics in general can be represented as shown in Figure 4. For a diagrammatic representation of the user interaction workflow refer to Figure 3. An overview of the steps in the user interaction can be described as follows:

![Figure 3: User Web Activity/Behavior Tracking](image)

1. The user sends a request to a primary web server.
2. The primary web server returns the web page along with embedded third-party JS reference.
3. Some client-side event causes an HTTP request to the third-party server that fetches the third-party JavaScript code and loads it into the DOM at the client-side.

4. This third-party code gathers the required information such as contextual keywords, client IP address, Browser Information, cookie value, etc. depending on what information the third-party server needs to track and sends it to the third-party server.

5. The third-party server processes the information collected and proceeds with the necessary further actions.

**Other examples**

1. Google AdSense: “Google uses its Internet search technology to serve advertisements based on website content, the user’s geographical location, and other factors[1].” Google AdSense technology is based on various mechanisms to provide the appropriate ads to the website for earning revenue. It gathers various kinds of information such as keywords of the website content, cached page, search results, etc. for contextual and customized ads. The third-party ad server (Google in this case) sends the necessary (based on website content) ad to the client, which is integrated and displayed to the user. The webpage returned by the primary web server contains a JS code that uses inlined JSON to fetched ads from Google’s server which is integrated into the primary webpage at the client side.

Google Analytics provide tracking in two different ways:

(a) Asynchronous Tracking
(b) Traditional Tracking

In both the cases, same tracking technology is used i.e. first-party cookies and JavaScript. The difference lies in the stage (during page rendering) at which the tracking script is executed and whether it is loaded asynchronously in the background when the page is being rendered or not. Below is the code for implementing asynchronous tracking.

```
1 <script type="text/javascript">
2 var _gaq = _gaq || [];  
3 _gaq.push(['_setAccount', 'UA-XXXX-X']);  
4 _gaq.push(['_trackPageview']);  
5 (function() {
```
2. Yahoo Web Analytics: Functionality and working similar to Google Analytics
3.7 P07 Geolocation Sharing

Overview

This pattern encompasses multi-party component interactions that involve sharing of user’s private information such as location with third-party components.

Pattern extends: P02

Wikipedia defines Geotagging as “process of adding geographical identification metadata to various media such as a Geotagged photograph or video, websites, SMS messages, or RSS feeds and is a form of geospatial metadata”[2] In simpler words, Geotagging can be defined as marking social media with location information. “Geocoded data has many applications, but until very recently, it was time-consuming and difficult for non-specialists to determine the physical coordinates of a place or object, and options for using that data were limited. Now, many common devices can automatically determine and record their own precise location and can save that data along with captured media (like photographs) or can transmit it to web-based applications for a host of uses. The full implications of geo-tagging are still unfolding, but the impact in research has already been profound.”[14]

Technical Pattern Description Various mashup scenarios can be observed that involve sharing of geocoded information. Geotagging implies sharing location information related to any kind of social media. Geotagging binds geocoded[11] physical location to any digital resource1. Not only can geotags be added to photos but also to websites, online statuses, online product shopping catalogs and any other digital media. This is achieved by adding latitudinal and longitudinal location co-ordinates as meta-information (tags) to the resource. Scenarios involving eHealth system can also be seen as the ones that gather location information. For e.g. patient’s geocoded information is stored by eHealth systems for patient monitoring and medical emergencies. The user interaction workflow for geotagging is represented in Figure 5.

As an example, an overview of the user interaction flow for Photo Geotagging can be described as follows:

1Geocoded information refers to geographic co-ordinates often represented as latitudinal and longitudinal values received from processing other geographic data such as postal code, street address, city, etc.
1. The client requests for a digital resource, a photo for example, from the web server.

2. The web server sends the photo along with its geo tags.

3. A third-party mapping service JavaScript is added to the DOM at the client side. There are variations of how this JS code is added to the DOM. Seen from most common examples, it can either be added by a user at the client-side for e.g. as a Greasemonkey Script in case of GMiF[3] or is returned by the primary web server as a part of the DOM for example in case of Flickr Geotagging with in-built Yahoo Maps. In any case, this third-party JS Script code refers to some map service such as Google Maps or Yahoo Maps as is loaded into the DOM.

4. A client-side JS event causes an HTTP Request to the third-party web server requesting for a map and map-markers with respect to the geocoded information.

5. At the client side, the resource and the corresponding marker are combined (or overlaid) with the map and rendered.

Figure 5: Geotagging Mashup
Many mashup services exist and can be easily implemented with variations that request digital resources (e.g. photos) from one third-party server, map from another third-party server and provide their own JS code that combines the two resources at the client-side. Such as JS code can utilize open APIs from both third-parties to overlay resources with their geotags onto the map.

Examples

1. eHealth Patient Monitoring
2. Flickr Photos Geotagging with in-built Yahoo Maps
3. Flickr Photos Geotagging with Google Maps in Flickr (GMiF)[3]
4. Petstore Mapping Mashup: A Petstore Application has a DB of meta-information (Geo Tags) assigned to the photos of pets that they want to sell based on where they are available for purchase
5. Housing: Houses on sale are mapped onto the Google maps, according to their Geo Tags, providing information about where they are located, sale-price, living area, supermarkets nearby, etc. All these information is represented as individual pins on the map.
6. Adding GPS Location to your Status messages in Google Plus and Twitter: Both Twitter and Google Plus allow adding Geolocation information to the status update.
7. Holiday Pictures
3.8 P08 Capability-based Authorization Delegation

Certain operations on assets owned by users are critical in nature due to the separation of trust boundaries between the communicating parties, sensitive nature of the assets, side-effect of the operation on the assets and flow of the sensitive information across different trust domains. In such situation a need arises to delegate the process of authorization. The authorization mechanism relies on a capability-based access model.

Pattern extends: P01

Technical Pattern Description The most commonly seen example today amidst social media technologies are the third-party applications integrated or plugged into a particular primary social media platform such as Android OS, Facebook, Flickr, Twitter, etc. These primary social media software for web or web-enabled devices provide their own APIs to develop third-party applications to provide certain integrated functionality to the users. Hence, comes into play, the need to manage authorization rights of the third-party applications to access user data or assets. The authorization is performed in a capability-based style. Delegation of authorization in such distributed work flows question and prove the legitimacy of operations performed by the third-party applications with respect to the user assets.

To understand the work flow it is vital to understand certain properties of the actors involved in the process:

1. The primary social media platform provides certain basic functionality and features to its users and manages user data and operations on it.
2. The primary social media platform also provides an API
3. The third-party application refers to an application, that has been developed based on the given API with the goal to provide additional features or functionalities to the user of the primary social media platform

An overview of the user interaction flow can be described as follows:

1. The user requests the third-party app to perform an operation on its asset managed by the primary social media platform
2. The third-party application sends the request containing the resource, operation pair to primary social media platform
3. The primary application delegates the authorization grant request to the user, requesting for granting necessary permissions for the third-party application.

4. In case the user is not logged on to his user account at the primary platform, he is presented with authentication challenge and logged in after providing a correct response.

5. The user sends back the necessary authorization permission, defining precisely whether the particular operation on requested user asset is granted or denied.

6. The primary application verifies the authorization permissions and if the access has been granted, it allows the third-party application to execute the necessary resource, operation request.

The workflow for the interaction pattern can be seen in Figure 6.
Examples

1. Cross-domain blogging: “Selective Tweets” for Facebook, “Twitter App” for Facebook, etc.

Twitter App for Facebook: It is a 3rd party application developed using Facebook API for posting twitter updates to user’s Facebook Profile. The Twitter App for FB has been developed using the Facebook’s JavaScript SDK[9]. Apps on Facebook.com are loaded in the context of Facebook.[7]. Access to user’s Facebook account information is restricted via access permissions. Refer to Figure 7 and Figure 8. The twitter app requests for access to user’s basic and profile information. Facebook API requires 3rd-party apps to authenticate via OAuth.

2. Twitter Apps: Applications developed using Twitter API
3.9 P09 Distributed Workflow

While performing various web activities, the user consumes different types of online services such as posting comments, buying articles, etc. For many such services, utilizing a 3rd-party service is a vital step in the successful execution of the primary service consumption.

**Pattern extends:** P01

In other words, the primary service provider consumes a 3rd-party service in order to successfully complete the current transaction in progress. The work flow of such a service consumption that involves a primary service provider and a 3rd-party service provider is distributed in nature. One of the steps in the primary work flow is extended to consume the 3rd-party service. That is, the basic service consumption work flow is extended to a distributed work flow involving communication with an external service provider. Needless to say, as the user serves as a consumer to the primary service provider, the primary service provider in turn serves as a consumer to the 3rd-party.

**Technical Pattern Description** Services such as online shopping, commenting on blog posts, etc. are based on distributed service management. While the primary service provider such as a blog server or an online shopping website is responsible for the management of the primary service, certain steps of the work flow require that the primary server contacts the 3rd-party server directly and/or indirectly (by redirecting the user to the 3rd-party server). The 3rd-party server provides service directly and/or indirectly to the primary server and thereby to the user. The interaction work flow is distributed in nature as depicted in Figure 9.

**Properties:**

1. The third-party service provider performs service while hiding sensitive user details from the primary service provider

2. The user is redirected to the 3rd-party thereby extending the control to include a 3rd-party trust domain

An overview of the user interaction flow can be seen in Figure 9:
**Examples**

1. **OpenID (Identity Management):** OpenID provides a identity validation service to other parties on behalf of the user. It provides the mechanism to authenticate users in a decentralized manner\[4\]. In brief, the user needs to validate its identity to the primary server to perform a certain action such as posting a comment on the blog successfully. In order to do so, a 3rd-party server that is implementing the OpenID protocol and serves at the OpenID Server, authenticates and validates the identity of the user to the primary web server without revealing any sensitive data about the user to the primary web server. The user shares a distinct relationship with both the service providers and therefore it becomes mandatory to keep the information shared with each of them within its trust boundaries.

2. **Paypal (Payment Management):** PayPal is an example of a CaaS (Cashier-as-a-Service) where the merchant implementing the online shopping workflow is responsible of providing the shopping service to the user. This work flow includes payment step which is performed ex-
ternally by consuming payment services from the 3rd-party, in this case PayPal. For further description refer to Work Package 2 Deliverable D2.1 Section 2.4 Online Shopping Workflows over Two Domains.

3. Extended work flow, Doctor acting on the behalf: In this scenario, the doctor acts on behalf of the patient to perform activities such as modify/creating/updating the patient’s medical records, managing medical bills in co-ordination with the patient’s health insurance provider.

The acting parties involved in this scenario are the patient and the doctor at the client side, the primary ehealth system and the 3rd-party insurance provider. The patient needs to provide the doctor with the necessary authorization rights to be able to access patient’s medical records and bills. This part of the entire scenario can be seen similar to the interaction pattern for the capability-based authorization. Next the doctor interacts with the ehealth system for managing patient processes. The billing step however takes place externally in a distributed manner with the 3rd-party insurance service provider.

Points to note: how to manage access rights to doctor, what information in ehealth system is accessible, what information is sent to the 3rd-party, what information created by insurance provider regarding the billing is accessible by the doctor
3.10 P10 Out-of-band distributed service

Pattern extends: P01

Technical Pattern Description  For increased Security, certain sensitive actions require explicit validation of the user. This is provided using a separate communication interface, e.g. a separate desktop application or an app on a mobile phone. The separate application queries for acknowledgement of the sensitive operation displaying the requesting-entity, the request in form of a text message and gives the possibility to authenticate and acknowledge or deny the request.

Figure 10: Out of band Distributed Services

The validation requests that are performed in this way may be used to implement separation of duties. Separation can be achieved with respect to sequence, which requires two authorization requests from the same user using different communication channels, and with respect to individuals, which requires two different authorized users and implements four-eye principle.

Figure 10 represents the steps of the user interaction flow. The work flow can be described as follows:

1. User sends a request to the primary web server to access some sensitive information.
2. The server requests the user for his primary credential to authenticate himself.

3. The user supplies his credentials (e.g., login/password combination) and is authenticated/verified by the server.

4. On successful authentication, the server sends a 2nd-factor authentication credential to the user via an out-of-band mechanism such as sending a PIN to user’s mobile device.

5. The user now sends this 2nd-factor credential value, that it received via an out-of-band channel, to the primary server for verification.

6. The out-of-band authentication can be based either on multi-factor credentials or multiple-users.

7. The server verifies the 2nd-factor credential value and confirms or denies user’s access request.

1. Multi-factor Authentication: The Google Authenticator project includes implementations of one-time passcode generators for several mobile platforms, as well as a pluggable authentication module (PAM). One-time passcodes are generated using open standards developed by the Initiative for Open Authentication (OATH) (which is unrelated to OAuth).

   These implementations support the HMAC-Based One-time Password (HOTP) algorithm specified in RFC 4226 and the Time-based One-time Password (TOTP) algorithm currently in draft[8]. 2-step verification adds an extra layer of security for your Google Apps account by requiring you to enter a verification code in addition to your username and password, when signing in to your account.

   Google Authenticator is a mobile application that allows you to generate 2-step verification codes on your smartphone without a network connection. We recommend users with smartphones to use Google Authenticator to generate verification codes instantly to sign in to their Google Apps accounts.

2. Mobile OTP
3. Mobile Transaction Numbers such as mTAN and smsTAN. The TAN number is sent to the user’s mobile and used as a 2nd step of authentication/authorization verification.
4 Usage Scenarios

To better understand the broad landscape of mashups and corresponding security requirements, a sample set of 15 small, relevant scenarios to be analyzed in WebSand has been selected. The scenarios are taken from real-world mashup systems found on the internet today and involve multiple stakeholders. They vary in size and complexity and are only briefly introduced in this section. A more detailed study will be done in WP2 (Secure Web Interactions), WP3 (Information-Flow Control) and WP4 (Secure Composition). The scenarios are nevertheless presented as an early set of representative scenarios to support the research discussion, drive the policy requirement studies and enable an early validation.

The small composition scenarios vary in size and complexity, and are only briefly introduced in this section. They are studied in more detail in WP2 (Secure Web Interactions), WP3 (Information-Flow Control) and WP4 (Secure Composition), and provide an early set of representative scenarios to support the research discussion and to drive the policy requirement studies, as well as enable early validation (in anticipation of the use case scenarios that are currently being developed in the course of WP1 and WP5). The following table lists the scenarios, related interaction patterns and references to other deliverables.
<table>
<thead>
<tr>
<th>ID</th>
<th>Scenario Name</th>
<th>Related Interaction Pattern</th>
<th>Related Deliverables</th>
</tr>
</thead>
<tbody>
<tr>
<td>S01</td>
<td>Google Maps</td>
<td>P01, P02, P04, P07</td>
<td>D3.1 Section 1, 2.1;</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>D4.1 Section 3.1</td>
</tr>
<tr>
<td>S02</td>
<td>Dangerous Goods</td>
<td>P01, P02, P04, P07</td>
<td>D3.1 Section 2.1</td>
</tr>
<tr>
<td>S03</td>
<td>User Tracking</td>
<td>P01, P06</td>
<td>D3.1 Section 4.1;</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>D4.1 Section 3.3</td>
</tr>
<tr>
<td>S04</td>
<td>Advertisements</td>
<td>P01, P02, P04, P06</td>
<td></td>
</tr>
<tr>
<td>S05</td>
<td>Tax/Loan Calculator</td>
<td>P01, P02, P07</td>
<td></td>
</tr>
<tr>
<td>S06</td>
<td>Third Party Libraries</td>
<td>P01, P02, P03</td>
<td>D3.1 Section 1.4, 2.1, 4.5</td>
</tr>
<tr>
<td>S07</td>
<td>Interactive Avatar</td>
<td>P01, P02, P04</td>
<td>D4.1 Section 3.4</td>
</tr>
<tr>
<td>S08</td>
<td>Third-party authentication</td>
<td>P01, P09</td>
<td>D4.1 Section 3.6</td>
</tr>
<tr>
<td>S09</td>
<td>Facebook Login</td>
<td>P01, P02, P09</td>
<td></td>
</tr>
<tr>
<td>S10</td>
<td>Third-Party Payment</td>
<td>P01, P09</td>
<td>D2.1 Section 2.4, 2.5</td>
</tr>
<tr>
<td>S11</td>
<td>Facebook Application</td>
<td>P01, P08</td>
<td>D4.1 Section 3.2</td>
</tr>
<tr>
<td>S12</td>
<td>Google Widget</td>
<td>P01, P02, P04</td>
<td></td>
</tr>
<tr>
<td>S13</td>
<td>Social E-commerce</td>
<td>P01, P02</td>
<td>D3.1 Section 2.4</td>
</tr>
<tr>
<td>S14</td>
<td>Contact Swap</td>
<td>P01</td>
<td>D3.1 Section 2.4</td>
</tr>
<tr>
<td>S15</td>
<td>Holiday Pictures</td>
<td>P01, P02, P04, P07</td>
<td>D4.1 Section 3.7</td>
</tr>
<tr>
<td>S16</td>
<td>Mobile Apps for Enhanced Security</td>
<td>P01, P10</td>
<td>D2.1 Section 2.1</td>
</tr>
<tr>
<td>S17</td>
<td>Delegation of Privilege in Distributed Workflows</td>
<td>P01, P08, P09</td>
<td>D2.1 Section 2.2</td>
</tr>
<tr>
<td>S18</td>
<td>Cross-domain Interaction</td>
<td>P01</td>
<td>D2.1 Section 2.3</td>
</tr>
<tr>
<td>S19</td>
<td>Web Portal to Send Limited Number of Text Messages</td>
<td>P01</td>
<td>D2.1 Section 2.5</td>
</tr>
</tbody>
</table>

Table 2: Scenarios and their relation to the use cases and the other deliverables.
4.1 S01 Google Maps

Google Maps (GMaps) is the most commonly used API in mashups [17]. In this scenario, the map functionality offered by Google is combined with application-specific data, such as geometric coordinates, points-of-interest, moving GPS coordinates, or custom overlays. This data is either directly available for the integrator, or is in its turn received for a third-party service provider. One particular instantiation of this scenario could be a client-side application that uses Google Geocoding API to translate a textual description of a location into GPS coordinates, and then uses these coordinates to place a marker on a map. Multiple queries can be performed in sequence and the previous markers are not erased. The Google Geocoding API and the GMaps API are commonly used as follows. First, the javascript API for the particular service is fetched via a script tag from the google domain. Next, the integrator code performs several queries via this service API to achieve the intended application logic. The combination of service API and client-side code can either run directly in the (outermost) document (i.e. mashup integrator document), or can be combined in a separate iframe.

In this scenario, multiple stakeholders are involved: the map service provider (Google), the integrator, and possibly also the data service provider and end-user. In case both the GMaps API and Geocoding API are included into the integrating page, giving them access to all contained functionality and datastructures. This implicitly places an enormous amount of trust in the providers of both APIs. Sensible security requirements would be to limit exposed functionality and datastructures between the gluecode and the different components.

4.2 S02 Dangerous Goods

This scenario considers a web application of a company that transports dangerous goods with trucks. A software system keeps track of where each truck is and what it transports. Using a mapping service like Google Maps, the locations of these trucks can be drawn on a map. But because the nature of the transported goods is dangerous, the mapping software should not...
be able to find out specific information about what each truck transports. This scenario is also discussed in the context of WP 3 as part of deliverable D3.1.

4.3 S03 User Tracking

Some website traffic analysis applications (like Google Analytics or ClickDensity) collect website usage patterns by tracking the interaction between a user and a webpage. Some metrics that can be used by this software including referer information, entry and exit point on a website, and various heat maps (hovering, dragging and clicking of the mouse, selecting, copying and pasting of text). In this scenario, the website developer wants to have some control over what the user tracking software is allowed to do and what it can monitor.

4.4 S04 Advertisements

Advertisements (ads) are commonly used by major corporations to convince potential customers to use their products or services. Ads are part of the business model of the early internet and are still widely used today. In contrast with the real world where advertisements are only displayed, ads on the internet are interactive. For instance, users can click on them and be taken away to a website with more information, an ad can expand itself and hide or cover the original website till it is clicked, or an ad can dynamically adapt to the hosting page (based on keywords, surrounding text fragments or even mouse movements or key strokes).

A commonly used pattern in advertising is to include a piece of third-party JavaScript code into a hosting webpage. When this JavaScript is executed, the hosting webpage in analyzed to determine what type of information is being displayed to the user. Using this information, an ad is selected that appropriately fits the context and is likely to attract the interest of the user. The ad is then displayed to the user in a format selected by the hosting
webpage. Typically, this has the form of a rectangular banner rendered at e.g. the top or the right side of the hosting webpage. Clicking the banner will lead the user to a webpage with more information about the product shown in the ad.

Displaying an ad involves several security stakeholders: the provider of the hosting page, the ad provider and the end-user. By serving an ad on a page through inclusion of JavaScript code, the provider of the hosting page places implicit trust in the ad provider. Sensible security requirements are then e.g. to only allow access to specific parts of the DOM of the hosting page, (See ADJail\cite{18}) and only allow interaction with a small set of third-party hosts.

### 4.5 S05 Tax/Loan Calculator

A client-side calculator that calculates a due tax or a loan based on the user’s financial data (bank savings, salary, real estate, ...). The salary or bank account information is considered sensitive data and should under no circumstances and in no form be leaked to the outside world, and even not to the calculator service provider.

### 4.6 S06 Third Party Libraries

A good practice on any software platform is to make use of reusable code. This is no different in web application programming. Several JavaScript libraries (e.g. jQuery, Dojo, ... etc) exist that contain premade software constructs and functionality. These libraries form a unique challenge because their functionality can not be easily isolated from the rest of a webpage; i.e. the potentially malicious or infected library code shares the same set of permissions as integrator code.
4.7 S07 Interactive Avatar

Most blogs and forums allow users to register an account in order to post comments. The account typically has the option to share information about oneself, like e.g. location, age, interests, ... in a user profile. This user profile usually also includes an avatar image that is displayed next to the user’s posts. The interactive blog scenario considers a user profile that allows users to create a more interactive avatar using JavaScript or a subset of JavaScript. Such an interactive avatar could for example be composed of embedded video/audio elements, or interactive canvas graphics.

The user could enter the code for this interactive avatar directly into a textbox, or more realistically, make a selection from a library of community-provided pre-made interactive avatars. Either way, a community user is allowed to input “avatar” code directly. When displayed, the JavaScript code is executed and can interact with both the hosting page and the visiting user. The main security stakeholders in this scenario are the webpage provider rendering the avatars, and the end-user. Optionally, the avatars could be hosted on a third-party site. In that case, the provider rendering the avatars places implicit trust in the avatar hosting provider. Care must be taken to limit the functionality available to the avatar code. However, enough functionality must still remain so that avatars remain usable and their usage is attractive to end-users.

4.8 S08 Third-party authentication

Using a third-party authentication provider is an easy way to integrate an authentication mechanism, and greatly increases the usability of the application (e.g. no new username/password combination for users to remember). A concrete example of third-party authentication providers are OpenID implementers.

A common way to use a third-party authentication provider is by delegating control of the browsing session to the provider. The provider will allow the user to authenticate in any of the supported ways (e.g. username/password, token, electronic ID, etc.). After having authenticated the user, the authentication provider will transfer the
control back to the originating site. The site can now continue loading the user profile, either with local data or data obtained from the authentication provider. In a mashup scenario, the third-party authentication provider is integrated as a component of the originating site. Instead of transferring control of the entire session, the control of only one part of the mashup will be delegated to the authentication provider. This ensures the continuous operation of the remainder of the mashup application.

4.9 S09 Facebook Login

In this scenario, Facebook APIs are used to facilitate authentication and authorization. By using the Login Button or Registration Plugin, Facebook helps website developers to simplify and enhance user registration and sign-in. Instead of maintaining a separate user database, website users no longer need to fill in yet another registration form or remember another username and password, but can sign in with their Facebook credentials. Facebook Platform uses OAuth 2.0 for authentication and authorization. As a website developer, you can add OAuth 2.0 support directly to the website, but more convenient, you can enable the authentication via a JavaScript API [9].

4.10 S10 Third-Party Payment

The services offered by third-party payment providers allow the easy integration of different payment systems in a web site, such as a web shop or ticket ordering service. By using a third-party payment provider, such sites avoid the complexity of dealing with sensitive data and operations, which is subject to strict regulations.

Third-party payment providers offer the complete interface for users to complete the payment. This includes account management, maintaining several accounts or credit cards and actually executing payment transactions. A site can integrate such a third-party payment service as an external component.
in the page (e.g. using an embedded frame) or redirect the user to the payment provider’s domain. When the purchase process reaches the payment phase, the site provides the payment service with the necessary details and delegates control to the payment service. After having completed the transaction, the payment service reports on the result and returns control to the originating site, which can further handle the transaction and continue the purchase process.

In this scenario, there are three security stakeholders: the original site integrating third-party payment services, the third-party payment service and the end-user. The integration of the payment service relies on the least-privilege principle. Delegating control back and forth results in legitimate cross-origin communication. Care should be taken that both the payment provider and the originating site are protected against non-legitimate cross-origin communication. Moreover, both have to secure their domains’ redirection and (re-)entry points respectively in terms of control-flow integrity.

4.11 S11 Facebook Application

Facebook applications are applications that make use of the Facebook API to retrieve information about Facebook users. This information (personal information, friend list, photos, messages, ...) is mashed with other data to provide added value to the user. In the past, Facebook applications could be created by third party developers in 2 ways: the application could be included on Facebook through an iframe, or it could be written in Facebook Markup Language (FBML). Since Facebook has deprecated FBML, we particularly concentrate on the former integration technique.

In the first case, the “iframe scenario”, the application is hosted on a third-party website and rendered in the browser in an iframe. Developers have full control over the application as if it were a separate web application. Because the web-application is in a separate iframe, it lives in its own realm and can not contact Facebook servers freely: it must obtain and use special user credentials to contact the Facebook API.

In the latter case, the “FBML scenario”, the application is written in FBML, stored on a third-party server and proxied through the Facebook servers themselves. FBML contains FBJS, a safe subset of JavaScript that does not allow the developer to perform insecure operations. For the Facebook ap-
plication in this scenario, it would have been interesting to see if, instead of writing the application code in FBJS and running it in an uncontrolled environment (the browser), the application could be written in plain JavaScript, and run in a controlled environment. The usage of such a controlled environment would allow more freedom to the Facebook application developer. In the iframe scenario, there are 3 security stakeholders: Facebook, the third-party application provider and the end-user. In the FBML scenario, there are 2 real security stakeholders: Facebook and the end-user. Facebook ensures that it offers secure code to the end-user by using FBJS.

4.12 S12 Google Widget

This scenario needs further checking!

iGoogle allows users to integrate widgets or gadgets on their iGoogle page and other websites. These widgets are rendered in an iframe and typically contain small pieces of information a user is interested in (like local weather, recent Twitter posts, ... etc.)

iGoogle gadgets are XML files which can contain JavaScript code, HTML content and other meta-data. When registered as an iGoogle gadget, the XML file is retrieved and processed by Google. This processing step extracts the code and HTML content from the gadget and, if the gadget developer chooses to, rewrites the JavaScript code with Caja. After processing, the gadget is hosted on its own subdomain of gmodules.com and will then appear in the iGoogle gadget directory for anyone to use. When integrated into an iGoogle webpage, the gadget is rendered in an iframe.

For a third-party iGoogle gadget scenario, there are 3 security stakeholders: iGoogle, gmodules.com and the end-user. Since the gadget is hosted in its own subdomain of gmodules.com and rendered in an iframe, several security problems are averted because of the same-origin policy. In its

2This domain is owned by Google and serves no other purpose but to host third-party iGoogle gadgets and prevent XSS against Google services. Google’s own gadgets are hosted on their own servers.
iframe, the gadget can still use all functionality at its disposal. If the gadget developer opted-in to Caja rewriting, the functionality of the gadget inside the iframe can be further limited.

4.13 S13 Social E-commerce

In this scenario, a e-commerce website and a social networking site cooperate together to grant a discount to shop users that recommend the store to their friends. The size of the discount depends on the price and a privacy-friendly approximation of the number of recommendations (i.e. friends that received the recommendation). For an end-user (as well as the social networking site), it is important that information about friends is shared with the shop gradually and in a privacy-friendly way. At the same time, the shop needs to be sure that valid information is provided to grant the discount. This scenario is discussed in more detail in the context of WP 3 as part of deliverable D3.1.

4.14 S14 Contact Swap

In this scenario, a user can let two web applications (e.g. Facebook and GMail) collaborate to synchronize profile contacts between each other. An important characteristic of this contact swap is that the import of contacts from profile A to profile B only proceeds if and only if at the same time the reverse flow of information is allowed as well. In other words, Facebook can import GMail contacts only if the Facebook contacts are offered to GMail for import as well. To achieve this, an environment should be created in which a fair exchange can take place, without cheating (e.g. the scenario where one of the applications is importing data, but not sharing anything in return, should be avoided). This scenario is discussed in more detail in the context of WP 3 as part of deliverable D3.1.

4.15 S15 Holiday Pictures
The holiday pictures mashup consists of three third-party components integrated with some glue code: a gallery component that stores the pictures, a photo editor component that can be used to edit the pictures and a map component used to draw geotagging information on a world map.

The components in this scenario combine a couple previous smaller scenarios. The map component is essentially the Google Maps scenario. Both the gallery component and the photo editor component can be seen as third-party gadgets.

In this scenario, there are several stakeholders. These are the end-user and 4 providers: the provider of the glue code, and the 3 providers hosting their own component. Interaction between the different components should be limited to a minimum. It makes no sense for e.g. the photo editor component to contact the map component directly. In addition, not all browser functionality should be available to all components: e.g. the map component should not have access to client-side storage functionality.

4.16 S16 Mobile Apps for Enhanced Security

In different scenarios, mobile phones are used as a trusted, secondary communication channel and as a second factor for authentication and authorization. The Google Authenticator[13] provides 2-step authentication using mobile devices and one-time passcodes. The passcodes are generated based on the HMAC-Based One-time Password (HOTP) algorithm[15] and the Time-based One-time Password (TOTP) algorithm[16]. Implementations are available as apps for Android and Blackberry phones and as a PAM-module (Pluggable Authentication Modules).

The two-phase process provides stronger authentication than the default password-based login for access to a Google account, all the more if performed on an untrusted device like a public PC, e.g. in hotels and Internet cafes.

4.17 S17 Delegation of Privilege
Mashups that involve back-end communication of mashup components must provide a mechanism for delegation of a set of access control privileges. An example of this delegation of user privilege is used by Facebook and Twitter and makes use of the OAuth protocol[10]. The OAuth protocol is designed to provide a simple and secure way for users of a web application, in our case Twitter, to grant access privilege to a third party, in our case Facebook, to their data and resources without forwarding their authentication credentials. In our example, the user wants that the posts he creates on Facebook should automatically also update his status on Twitter.

4.18 S18 Cross-domain Interaction

In the recent years on-demand solutions such as SAP’s Business By Design are becoming more and more important. By being provided over the internet these solutions offer companies more flexibility when designing their internal IT systems. Instead of buying and installing software on-premise, the on-demand solution can be rented for a certain amount of time and be accessed through a web browser. As information systems often need to communicate with each other, a communication channel is needed between on-demand and on-premise systems. Often, however, there are multiple network barriers such as firewalls between those systems. Hence, alternative communication channels such as cross-domain requests are used in order to connect remote systems.

In the following, we consider an on-demand application that provides a feature to display a personalized catalogue, in which the customer can view items that are currently running low or are out of stocks in his warehouse. Furthermore, the customer is able to receive recommendations for items that might be of interest to him. In order to gain the necessary warehouse data the on-demand system queries the customer’s on-premise warehouse system via client-side cross-domain requests through the user’s web browser. In order to enable this indirect communication channel the on-premise system has to open up parts of its API to client-side cross-domain requests. This can be accomplished by several technologies such as Adobe Flash, Silverlight or Cross-Origin Resource Sharing.

In order to support old legacy browsers as well as modern mobile browsers, the several different approaches have to be combined. So the server (here the on-premise system) has to setup multiple different policies for different environments. As each policy format is very different and as there are multiple security pitfalls in each technology, it is tedious to manually maintain these policies in a secure fashion. Websand will address this issue by offering an...
automatic way to maintain the different policies while avoiding the security pitfalls.

4.19 S19 SMS Portal

After providing the recipients’ phone numbers and the respective text, an HTML form is sent to the provider’s gateway in order to cause SMS sending. However, sending of more messages is denied when the maximum number of messages is consumed. The provider stores the respective amount of sent messages in a database.

Malicious customers could try to send a large number of requests at the same time to exploit a race condition, and, thus, send more messages than actually allowed.

There are numerous providers for sending free SMS. In fact, they are mostly localized. Some of these providers require an account, while others allow sending messages after consumption of ads.
5 Use Case: e-Health

The WebHealth application represents the e-Health use case of WebSand and incorporates a set of challenging work flows and usage scenarios in the domain of electronic health. The central object of WebHealth is a management system for electronic health records described from the perspective of patients and general practitioners, which includes computer-aided medical decision support, patient monitoring using portable monitoring devices and patient consent. Because of the complexity of real electronic health systems, a subset of system aspects has been selected that is still manageable in the scope of WebSand but nevertheless includes challenging usage scenarios and interaction patterns. In the following sections, the application domain electronic health is introduced, the system architecture of the WebHealth system is described and the mashup relevant workflows are described.

5.1 e-Health Application Domain

In the following, an overview of the e-Health application domain is given, in which the e-Health use case is located.

eHealth (or electronic health) is broadly defined by the World Health Organization as the 'use of information and communication technology for health' [19]. The objective is to use Information and Communication Technology to improve health care service delivery through the strategic use of technologies such as computers, Internet, satellite receivers, and personal mobile devices.

As the Unit H1 of the European Commission (ICT for Health Unit) describes in the Portal 'What is eHealth?' [12] 'eHealth covers the interaction between patients and health-service providers, institution-to-institution transmission of data, or peer-to-peer communication between patients and/or health professionals. Examples include health information networks, electronic health records, telemedicine services, wearable and portable systems which communicate, health portals, and many other ICT-based tools assisting disease prevention, diagnosis, treatment, health monitoring and lifestyle management.'

The two main technological innovation areas regarding eHealth are (see for instance the scope of the journal I. J. Medical Informatics, from the European Federation for Medical Informatics (EFMI), European Regional Council of the International Medical Informatics Association (IMIA)):

- Computer-aided medical decision support systems based on heuristics, algorithmic and/or statistical methods (decision theory, artificial intelligence, data mining, analysis of images, etc), and
• Information systems, including national or international registration systems, hospital information systems, departmental and/or physician’s office systems, document handling systems, electronic medical record systems.

Most security problems arise in information systems that collect, manage, handle or store personally identifiable information (PII) patient data. We call any such record an electronic Health Record.

Manufacturers and care providers experience a is a strong pressure to create a flexible information infrastructure that helps in providing more efficient methods in health care, public health, and related areas. Health ICT should

• make available to care personnel the patients health records (test exams, current diagnoses, allergy information, medical history, etc) in the moment they need it,

• support clinical decision making for the prevention and treatment of illnesses,

• be used to acquire population data for public health purposes like epidemics and distribute the results,

• support the communication between the health care providers, third parties, and patients, and

• help improve the data quality for research purposes.

Both the lack of information about the medical history of a patient and the unavailability of medical guidelines or decision support has been the cause of improper diagnosis or treatment.

It is therefore not surprising that eHealth is an area of rapid innovation. Many different solutions have been advanced and will continue to emerge, both for information systems, which manage patient data or other medical information, and for computer-aided medical decision support systems. And although the proposals have different (and often proprietary) formats, models, systems, with the resulting interoperability problems, several of them will co-exist for at least the next couple of decades. There exist several proposals for standard formats, exchange protocols, and access control models for electronic personal health information, but there is still no global consensus. There is a strong need to integrate them in a coherent way.
5.2 System Architecture

The WebHealth system represents a set of challenging work flows and usage scenarios in the domain of electronic health. The central object is a management system for electronic health records from the perspective of patients and individual healthcare providers (doctors). A third-party that is also involved in the workflows is a medical database, which gathers statistical data and provides this data as a reference to support medical decisions. The WebHealth system provides two mash-up user interfaces, one for the patient and one for the doctor, for access of two back-end systems for management of electronic health records and a public database with medical reference data. The following list of key aspects of the system are subsequently described in detail:

- Gathering of long-term measurements and patient monitoring using portable monitoring devices.
- Support of medical decisions using medical reference data from a public database.

Users

**Patient**  The patient represents the medical subject who’s medical condition is the main concern of the health system. Associated to the patient is
his medical history containing individual health records. The patient receives health services from health care providers, e.g. a doctor.

In general, the patient controls the data concerning his medical status and makes independent decisions concerning medical treatment. For example, he chooses his doctor, or he may refuse a treatment. In some cases, the patients privileges are delegated to a third-party, for example to a child’s parents.

**Doctor**  The doctor is an individual health care provider that, among other services, offers consultation, conducts examinations, diagnoses diseases, and prescribes medical treatments. He may access public databases with medical reference data to support his medical decisions.

Additional medical staff members, for example nurses, and administration staff members, for example receptionists, may be employed by the doctor, which also interact with the WebHealth system.

**Systems**

**Patient Frontend Mashup**  The patient frontend mashup is the primary, web-based user interface for the patient, which allows management of patient-doctor-relationship, access to the patient’s medical history of electronic health records and management of the patient’s master data, e.g. his living address.

The patient frontend mashup represents the combination of mashup components, which originate from different server systems, running in the web browser environment.

**Doctor Frontend Mashup**  The doctor frontend mashup is the primary, web-based user interface for the doctor, which allows management of patient-doctor-relationship, management of the medical history the doctor’s associ-
ated patients and management of the doctor’s master data, e.g. contact information.
The doctor frontend mashup represents the combination of mashup components, which originate from different server systems, running in the web browser environment.

**Web Mashup Server**  The web mashup server is the server system that hosts the doctor frontend mashup and the patient frontend mashup web applications.

**EHR Server**  The EHR server manages the electronic health records (EHR) that comprise the patient’s medical history. It stores data related to examinations, imaging, diagnosis and prescriptions along with associated meta-data, e.g. regarding ownership and access control privileges.

**Master Data Server**  The master data server is responsible for management of user accounts for patients and doctors as well as management of the portable monitoring devices and the patient-doctor-relationship.

**Reference Data Server**  The reference data server represents an organisation that provides medical reference data to health care providers to aid in medical decision processes. In our workflows, a service offers reference data for electrocardiograms of healthy subjects and of subjects with a heart disease.

**Data Objects**

**Electronic Health Records**  The primary electronic health records that are managed by our system is information that identifies individuals and elements of their medical records, for example:

1. given name and surname
2. gender, blood type
3. date of birth, date of death
4. blood pressure, heart beat frequency
5. body weight, height
Some data elements, like the blood type, do not change for one medical subject and are managed as attributes. Others, like the blood pressure, are the result of a medical examination at a specific point in time, and these are managed as part of the medical history.

**Medical Reference Data** In the WebHealth system, the doctor is able to access medical reference data from a public database to aid in his medical decisions. Measurement data obtained in examinations can be compared to reference data of healthy subjects and subjects with specific pathological conditions.

**Patients’ Geo Location** In the monitoring workflow, the patient’s vital’s are monitored using a portable device. The examination data is transmitted in real-time together with the patient’s current location for emergency cases.

### 5.3 Workflows

![Workflow Diagram]

Figure 14: The consultation process and the workflows in the consultation process.

The general consultation process that is modeled in the WebHealth system is divided into three workflows that incorporate different mashup characteristics and interaction patterns, which are subsequently described. Figure 14 shows the overall process and the workflows in the consultation process.

- The patient requests medical advice and authorizes the doctor to access his electronic health records.
- The doctor accepts patient and schedules an appointment for the consultation and possible medical examination.
- An examination is performed and the examination data is stored as an electronic health record in the medical history of the subject.
- The doctor evaluates the examination results and stores his diagnosis in the medical history of the patient.
- After discussion with the patient a treatment plan may be stored in the patients’ medical history.
Patient-Doctor-Relationship

The patient-doctor-relationship workflow is about establishing the patient-doctor-relationship.

**Actors** The actors of this workflow are the patient and the doctor.

**Preconditions** An established patient-doctor-relationship must not be present.

**Description**

- The patient initiates a consultation request and asks to be accepted by the doctor.
- The doctor reviews the patient’s information, accepts the patient and schedules an appointment for the consultation and possible medical examination.

**Patient Monitoring**

The patient monitoring workflow represents a situation where the doctor collects the patient’s vitals over a period of days using a portable monitoring device, for example an ambulatory electrocardiography device. During that time, the patient’s geographic location is tracked to allow for medical support in case of emergencies. Figure 16 illustrates the patient monitoring workflow.
Figure 16: Mashup components and interactions in the decision support workflow.

**Actors**  The primary user or the actor in this workflow is the doctor. The goal of the doctor is to monitor the patient’s medical condition.

**Preconditions**  The doctor and the patient have an established patient-doctor-relationship.

**Description**

1. The doctor sets up the device and associates it with the patient.

2. While the patient wears the portable monitoring device, the device is measuring the patient’s vitals and continually transmits the data to the EHR server together with the patient’s current geographic location.

3. During the long-term measurement period, the doctor periodically checks the transmitted data.

**Medical Decision Support**

To come to a diagnosis and to decide on a treatment plan, the doctor examines the collected measurement data in the medical history of his patient and compares it with reference data from a public medical database. Supported by the reference data from the database, the doctor stores a diagnosis and proposes a treatment plan to the patient.
Figure 17: Mashup components and interactions in the decision support workflow.

**Actors**  The primary user or the actor in this workflow is the doctor.

**Preconditions**  The doctor and the patient must have a patient-doctor relationship, and measurement data for diagnosis must be available in the medical history of the patient.

**Description**

1. The doctor accesses the patient data using the diagnosis page.
2. The doctor enables the diagnosis decision support feature and selects the desired data source.
3. The reference data is retrieved from the reference data server and displayed in a view in comparison to the patient’s health information.
4. After reaching a conclusion, the doctor stores a diagnosis and proposes a treatment plan.
6 Use Case: Sales Everywhere

The Sales Everywhere solution is a sales software designed for small and medium companies that want to focus on their primary task, sales. This is achieved not only by providing the software and the underlying hardware but by hosting the systems too. Therefore, all technical aspects of the IT-landscape are taken over by the software supplier. The advantages are numerous: updates or new softwares can be deployed nearly instantly, the system can be easily scaled depending on the customer’s needs... This architecture consists of different actors with different trust levels and boundaries that will be described in the next section, making it a good candidate for its security aspect.

6.1 Application Domain

The Sales Everywhere solution provides an efficient and attractive tool that can be used through the whole sales process, from defining the leads to the final sales order and delivery to the customer. This product is made for the sales representatives and the way they work, helping them with their needs to access various and real-time data and with the collaboration aspect.

![Figure 18: Illustration of the Sales Everywhere process.](image)

The main steps of the sales process are illustrated in Figure 18 and can be described as follow:

- Leads: the process starts with the search for potential customers or so-called ‘leads’. It can be someone who expressed his interest by registering to the company website or visiting the company’s stand during a trade fair.

- Opportunity: once a potential customer expresses a real need, an opportunity is then created.

- Quotation: a quotation is a first offer made by the vendor, a base for discussion between him and the customer.

- Sales Order: after mutual agreement, a sales order is created and the good is finally delivered to the customer.
6.2 System Overview

In the following sections, the focus will be on the ‘opportunity’ and ‘quota-
tion’ phases. The different aspects will be detailed in the workflow section 6.4 but the main steps from a sales representative point of view are:

- Getting the customer’s needs
- Presenting the results
- Making a cost estimation

The purpose of this system is to facilitate the daily work of a sales representa-
tive by providing an easy and secure access to the customer’s data (require-
ments, current inventory), its own data (customer’s history) and partner’s data (product details like availability and localization).

6.3 System Architecture

Fig. 19 gives an overview of all involved parties.

![Diagram](image)

Figure 19: Overview diagram of the main actors in the Sales Everywhere use case.

Primary Users

**Customer** The customer is a human agent who wants to buy a product offered by the vendor. In general, the customer provides a logistic system in his own network with a list of products he needs and eventually a list of all the products he currently has with their availability.
Vendor The vendor offers products (his own or partner’s products) to the customer. The software and hardware used for the sales process are under the control of a service provider. In certain cases, the vendor may own an ERP\textsuperscript{3} system in his network connected to the service provider’s server. In this system, the vendor can for example keep the shopping history of his customers or customize business processes. The sales representative works for the vendor and is responsible for the sales process at customer’s site.

Third Parties

Service Provider The service provider develops the Sales Everywhere solution and makes it available for the vendor. He also provides the servers where the software is installed and is in charge of the maintenance tasks.

Partner The partner can either provides the goods that will be offered by the sales representatives and/or software add-ons that will be installed on the service provider’s server. Those add-ons usually offer new or enhanced functionalities that are not part of the Sales Everywhere solution.

Web-Services Third parties web-services like geolocalisation or social networking services may be used. Those services can be integrated in order to enhance the catalogue by adding the current position of an item or the number of people who ‘like’ an item.

Data Objects

The Sales Everywhere solution manages different objects from various origins: it can receive a list of goods from the customer’s system, a customer identification file from the vendor’s system, an item file from the partner or even executable code from the partner.

Item An item represents a product that is sold by the Sales Representative. An item is characterized by attributes like description, price, availability... Items can grouped together to form a wish list (list of items the customer is interested in), an inventory list from the customer (list of items the customer already has) or an inventory list from a partner (list of items the partner offers). In all the above mentioned cases, the list will be sent to the vendor.

\textsuperscript{3}Enterprise Resource Planning
Customer Details File  For each customer, the sales representative keeps a file containing information like name, billing and shipping addresses... In addition, the list of all previous transactions is available. With this information, the sales representative is able to make a personalized offer to the customer.

Sales Everywhere Extensions  The Sales Everywhere provides extension points where a partner can insert his own code and develop new functionalities. The source code is then part of an add-on that can be bought by the Sales Representative and the add-on will be installed on the service provider’s server.

6.4 Workflows

Getting the customer’s needs

The sales representative is now at the customer’s site and wants to know what the customer needs and eventually propose other items he may be interested in (Fig. 20).

1. With his own mobile device (laptop, smartphone...), the sales representative uses the customer’s network to log in to the service provider’s server.

2. The sales representative then connects to the customer’s system that contains the list of items the customer needs. Due to possible network barrier, the server may not be able to communicate with the customer’s system, therefore the need for a client-side cross-domain request.

3. The sales representative (and the Sales Everywhere solution) is finally able to read what the customer needs.

Presenting the results

The Sales Everywhere searches for items that match the customer’s request and the results are presented as a catalogue with the items that match his requirement at the top, followed by other items he may be interested in. As the sales representative has access to the customer’s system, he can check if the customer already has an item before suggesting it.

The catalogue is not static, it can fetch content from 3rd party services like the Facebook ‘I Like’ button that displays how many Facebook users like an item or a geolocalisation service like Maps, so the customer can see where the items are currently located (Fig. 21).
1. A list of matching items is generated by the Sales Everywhere solution.
2. For each item, availability and price are checked by connecting to the suppliers’ systems.
3. For each item, additional content can be retrieved using 3rd parties web-services (client-side mashup).

Figure 21: Presenting the results
First Offer

Once the items have been identified, the sales representative connects to his own ERP system through the Sales Everywhere server to come up with a first offer. The price is then calculated depending on the customer’s history (number of previous orders, total amount spent...). The sales representative has the possibility to define complex rules to calculate the discount in its own ERP system (Fig. 22).

1. The vendor logs in to his own ERP system through the service provider’s server

2. A price is calculated using rules defined by the vendor (shopping history, amount of items...)

As the vendor’s system and the service provider’s server are located in a trusted zone, the implementation of this workflow in WP5 will be optional.

Customization of the Catalogue

If the sales representative is not satisfied with the default design or would like to have extra features not proposed by the service provider, he can request the installation of add-ons developed by partners. Those add-ons will run on the service provider’s server and enhance the default catalogue.

As the partner may not be familiar with the programming language and infrastructure used by the service provider, a new easy-to-use scripting language has been developed. The partner can develop his add-on on his own...
system using an IDE (Integrated development environment) also provided by the service provider and then send it to the service provider. The code will then be translated into the service provider’s programming language and the add-on will be installed on the Sales Everywhere system (Fig. 23).

1. A partner develops an add-on on his system using the new scripting language

2. The add-on is then translated by the service provider in a compatible add-on with the actual Sales Everywhere solution

3. The add-on is installed by the service provider on his server.

4. The sales representative can use the new functionalities

Figure 23: Customization of the Catalogue
7 Conclusion

Interaction Patterns provide a template, a mechanism or blueprint to understand and study the interaction flow, related dependencies, functioning and interaction between multi-party components experienced in today’s secure web interactions.

A pattern catalog is provided that delineates a handful of important interaction patterns that have been carefully studied, gathered and selected to cover the various scenarios and use cases gathered by the industrial partners SAP and Siemens in work packages WP2, WP3, and WP4. We have also provided two use case scenarios viz. “eHealth” in Section 5 and “Sales Everywhere” in Section 6.

The concept of Interaction Patterns is introduced with respect to multi-party web interactions in section 1. Section 2 sheds further light on detailed description on Interaction Patterns. The section determines and specifies the description of the Interaction Pattern as “a structured, generic description of a frequently found interaction between two or more mash up components or multi-parties components and actors”.

These Interaction Patterns have been derived based on the challenging scenarios that have been selected by all partners as a representative set of future internet applications from the perspective of the main problem fields of mash-up security, which are addressed by the technical work packages: WP2 Secure Web Interaction, WP3 Information-Flow Control, and WP4 Secure Composition.
References


