m:Ciudad: enabling end-user mobile service creation

Marcin Davies
FTW Telecommunications Research Center Vienna, Vienna, Austria

François Carrez
Centre for Communication Systems Research (CCSR), University of Surrey, Guildford, UK

Juhani Heinilä
Technical Research Centre of Finland (VTT), Tampere, Finland

Anna Fensel
FTW Telecommunications Research Center Vienna, Vienna, Austria

Maribel Narganes
Telecom Business Unit, Tecnalia, Zamudio, Spain, and

José Carlos dos Santos Danado
Department of Computer Science and Information, Norwegian University of Science and Technology, Trondheim, Norway

Abstract
Purpose – Mobile computing enables end-users to create small services on their mobiles and share valuable and context-aware information with others. The purpose of this paper is to introduce a platform for end-user generated mobile services – so-called microservices.

Design/methodology/approach – As a key component the authors present a microservice description language for user-driven mobile service creation and platform-independent service execution and rendering. The paper also gives insight into the authors’ visual authoring tool. The chosen design approach is evaluated in two phases: an intermediate evaluation with a small hands-on trial and an online survey; and a final laboratory test with 24 test users in total.

Findings – The paper provides empirical insights about the methods and motivations of end-users creating small mobile services. The main purposes of service creation would be mostly to exchange information, stay in contact, and just for fun (on the basis of non-commercial use). The evaluations also indicate the visual drag and drop approach of putting service blocks together as being the most favored in terms of user satisfaction.

Originality/value – The concepts and findings introduced in this paper will help in designing mobile service authoring environments, which is appealing to software communities/vendors and mobile network operators. The presented platform is, to the authors’ knowledge, the first designed and implemented infrastructure enabling end-user mobile service creation.

Keywords Mobile technology, Mobile communication systems, Semantic web services, Microservices, Mobile services, End-user development, User-centric service creation

Paper type Research paper

This work has been supported by the European Commission, ICT FP7 Collaborative Project m:Ciudad – A Metropolis of Ubiquitous Services (www.mciudad-fp7.org).
1. Introduction

The strong growth in mobile phone usage and increasing mobile computing power offers huge opportunities for new mobile services and applications. Current trends in the so-called Web 2.0, where users are not only consumers but also producers of content, could be moved to the mobile world, enabling mobile end-users to become active actors in the process of service generation and provisioning. Furthermore, the potential of mobile users as service providers is even higher in the mobile context as mobile devices are always attached to users, offering an always-on open door to the world.

With this idea in mind, we have designed a new infrastructure for user-generated mobile services with the mobile device as the central component. Small mobile applications, so-called microservices, that are focused on specific user needs, can be created “at the point of inspiration”, from the mobile itself, and also executed and consumed on the mobile device, at any time and anywhere.

Myriads of knowledge[1] generated by millions of potential users can be made available through this kind of services. Microservices will allow users to obtain and provide semantically enriched content, like opinions and recommendations, to fellow users.

With this purpose, our work defines an architecture, a set of mobile tools and a set of supporting components at the network side, to allow mobile users to create small, sharply focused, and knowledge-based microservices.

Some examples of usage scenarios for microservices are:

• **Mobile blog.** A user decides to share his experiences on-the-fly and sets up a micro-blog service that allows him to create multimedia blog entries and to provide them to his colleagues; each blog entry includes multimedia content (a text, a picture, a video, and an audio recording) describing a particular moment in life; furthermore, users accessing blog content will have the chance of adding comments to each particular entry.

• **Friend locator.** A group of friends meet every Sunday to watch football. Each of them goes to the stadium separately and the first reaching the ticket office has to buy the ticket for all. For this purpose one of them decides to create a friend locator microservice that allows him and his friends to share their location and show it on a map; all of them activate the microservice one hour before the match begins and know the positions of the others and of course who is reaching the stadium first.

The main objectives of this paper are to explain the infrastructure, the microservice description language as a key enabler for user-generated services, and the results of the user evaluations performed. Consequently, this paper is structured as follows: in Section 2 we provide an overview of the general architecture, the language for end-user driven microservice creation and its role within the service framework. The microservice description language is discussed in detailed in Section 3. Section 4 gives insight into the visual authoring tool and Section 5 discusses the user feedback gathered in early evaluations. Section 6 presents the results of the final evaluations. Related work is considered in Section 7, while Section 8 concludes the paper and gives an outlook on future work.
2. Overview of the microservice platform

As explained in the introduction, microservices are small and sharply focused services, which are designed by end-users on their mobile phones. Microservices can be used to share pieces of context-based content (opinions, recommendations, pictures, audio, etc.) that is collected either automatically, by terminal capabilities, or manually, by direct user input. This section introduces the overall platform and the microservice description language.

2.1 Architecture

The microservice architecture is composed of the following key components:

- **The service creation kit (SCK).** A mobile application used to create new microservices from scratch or to customize existing microservices. The SCK offers different user modes to accommodate different experience levels: basic, intermediate and advanced. The SCK is described in detail in Section 4.

- **The service execution environment (SEE).** A component, which is responsible for the execution of microservices on the mobile device. It also handles the requests coming from microservice clients. Finally, the SEE also manages the interfaces to the terminal capabilities used when creating a microservice in the SCK. More detailed information can be found in Section 3.6.

- **The service warehouse (SWH).** A network-side component responsible for the management of the microservices in the m:Ciudad framework. In particular this component stores most of the information attached to a microservices (as a bundle). Newly created services are uploaded from the terminal to the SWH so that they can be visible within and searchable by the m:Ciudad user community.

- **The knowledge warehouse (KWH).** A network-side component that manages and stores the semantic aspects of microservices such as information contained in the service profile and the content description. The KWH manages a number of ontologies such as domain and content ontologies. More information on this can be found in Section 3.2.

- **The search and recommendation engine (SRE).** This component allows for searching the eco-system of microservices available within the m:Ciudad framework. It also gives proactive recommendations to the mobile user. Of course, the SRE relies heavily on the semantic descriptions stored in the KWH and returns all service instances from the SWH (sorted by relevance) that fit the search criteria expressed by the end-user.

Whereas the SCK and SEE are terminal-side components, the SWH and KWH are located in a central place in the network side. The exchange of content is peer-to-peer (there is no knowledge or content stored in the network), but the architecture is centralized as far as the repository (SWH) and registry (KWH) are concerned (this is a design choice made, future work will also investigate a fully decentralized infrastructure).

At the terminal side, the SCK works with so-called simple service elements (SSEs) as building blocks that provide the user with an abstraction of the underlying language and capabilities (Sections 3.3 and 3.4). Those SSEs can be mapped to capabilities, residing either in the terminal or in the network, that are invoked at execution time by the SEE (as explained above). The SEE is essentially an embedded mobile web server that allows remote HTTP connections from other terminals[2] and also executes the microservice logic.
At the network side, the search engine is the main component linking the SWH and the KWH. It accepts search requests for:

- microservices and related content relevant for a user in a given context;
- microservices that users can modify to make their own services; and
- SSEs, service building blocks that can be used for service creation.

The general (and simplified) microservice lifecycle is shown in Figure 1 (Davies et al., 2008). At a first stage, users will create services by combining the previously mentioned SSEs or by customizing an existing service. Microservices will then have to be deployed in order to make them searchable and executable in the embedded SEE of the users’ terminals (it should be noted that we distinguish between providers of a service and simple users, i.e. consumers that just access a microservice).

2.2 UDL: microservice description language

As already motivated in the previous section, a description language for microservices is a central contribution of our work. This section gives an overview on the UDL – the U+ Service Description Language[3] – and its role within the system architecture.

Microservices are processed by an execution environment running in the mobile terminal (Section 3.6). Therefore, they need to be specified in a language that is:

- easily processed in a mobile terminal, and with direct access to terminal capabilities;
- easy to create, i.e. a language that can be mapped to higher level building blocks understandable by the user that creates the service; and
- highly expressive, so that a wide variety of microservices can be specified using the language.

Semantic descriptions play an important role in achieving these objectives. They are needed to characterize what the microservice is about and what kind of content it provides. Moreover, they enable much more precise microservice discovery by other members of the community and an easier and more accurate microservice creation process by annotation of service building blocks (SSEs).

2.2.1 UDL in a nutshell. The UDL is split into five sub-languages, which are briefly introduced in the following and shown in Figure 2:

1. **Microservice profile.** The UDL service profile (UDL-SP) provides a descriptive and functional definition of the service (i.e. “what the service is doing”). Essentially, the UDL-SP is an ontology with a set of metadata that is ultimately used by the search engine in order to discover microservices that match specific semantic properties similar to the service profile ontology of OWL-S (Martin et al., 2004). These properties can be expressed either with regard
to the microservice ontology (e.g. “find all services of type blog”) or a content ontology (e.g. “find all services dealing with a particular kind of content or find any microservice created by James”). This content ontology (UDL content description – UDL-CD) is explained shortly below and in more detail in Section 3.2. More information on the microservice profile is given in Section 3.1.

(2) Microservice content description. The UDL-CD considers the definition and the properties of service content (content is what any microservice is ultimately producing or sharing). Content descriptions are also ontology based and rely on existing domain ontologies. They are discussed in more detail in Section 3.2.

(3) Microservice capability profile. The UDL capability profile (UDL-CP) provides semantic descriptions of SSEs (in an ontology) for ensuring a quicker and more accurate creation process as only blocks that can be linked to existing blocks (in terms of IOPEs[4]) would be proposed to the user in the SCK. The UDL-CP is explained in detail in Section 3.3.

(4) Microservice logic. The UDL service logic (UDL-SL) gives an operational definition of the microservice. A formal definition of the elementary operations within a microservice – the previously mentioned SSEs – is necessary for service execution (by the SEE). The execution of a microservice results in either:
the creation/modification of content by the microservice provider;
the access and/or modification of content by a microservice user; or
the exchange of content between microservice users.

With regard to the service creation paradigm the project pursues a block-based approach, where users are able to specify the functionality of a service (Section 3.4). In any case, running a microservice results in the execution of elementary steps which have to be defined precisely in the UDL-SL (hence, this task is close to the traditional definition of a programming language). The UDL-SL definition is an XML-based language, which borrows from the WS-BPEL specification (Barreto et al., 2007). More information can be found in Section 3.4.

(5) Microservice rendering. The UDL service rendering (UDL-SR) considers the rendering of microservices on a mobile terminal. Because mobile terminals can vary in nature and characteristics an abstraction layer is needed that generally describes how each elementary piece of a microservice is rendered on the user’s mobile terminal. Therefore, the UDL-SR language proposes ways to define all rendering objects (i.e. user interface elements) and aspects associated with the service logic at execution time. The UDL-SR is discussed in detail in Section 3.5.

In conclusion, these five sub-languages/ontologies are used to describe microservices and the SSEs they are made of. A number of templates and ontologies are used at various places in the (simplified) architecture, which is shown in Figure 3. The purpose of the next section is to explain how and where the individual parts of the UDL are used.

2.2.2 UDL within the architecture. The UDL itself is not part of the architecture as such, but the templates and ontologies which result from a microservice, its content and all SSE descriptions have to be stored and used within the architecture. The UDL-SL (describing service logic) and UDL-SR (for the service presentation or rendering) templates are produced by the SCK and run by the SEE. It is worth noting that reverse engineering is enabled, which means that it is possible to re-build the original graphical view from the UDL-SR and UDL-SL templates within the SCK (Section 4). This property is essential to guarantee that a user can download an existing microservice, load it into the SCK and customize it, thus reducing the efforts needed for service creation as more microservices become available in the framework.

The service profile ontology used for the UDL-SP describes what the microservice does. All metadata is expressed with regard to this ontology. Metadata elements are used and stored at various places within the architecture (Figure 3):

- The SEE works with metadata elements that influence the way a microservice is executed locally, i.e. operational metadata (subset of UDL-SP translated in XML).
- The SWH processes metadata that influence the way the service is published, accessed, and viewed by others (subset of UDL-SP translated into XML).
- The KWH stores and uses the metadata elements that influence the way a service can be discovered (the whole UDL-SP in its native form, i.e. the form resulting from the SCK, which can also be called “internal form”).

Storing and managing microservice descriptions is the main function of the SWH and KWH, which can be seen as an analogy to the concepts of repository and registry within
SOA, respectively (Bell, 2008). The role of the SWH is the repository (i.e. the place where microservices are stored and from/to where they can be downloaded/uploaded), whereas the KWH plays the role of the registry (i.e. the place where microservices are published and described, and where lookup functions take place).

The semantic descriptions of the SSEs (UDL-CP, Section 3.3) and of service-specific content (UDL-CD, Section 3.2) are also stored in the KWH and can be accessed by the search engine. While the service profile and the content description are of primary interest for end-user search requests, SSE descriptions are actually accessed by the SCK during the service creation process (Section 3.3 and 4).

3. Microservice description language elements
As introduced in Section 2.2, the microservice description language UDL consists of five sub-languages/ontologies, which are discussed in more detail here. We also describe the actual execution aspects of the UDL in Section 3.6.

3.1 UDL-SP – microservice profile
The service profile provides a descriptive and functional definition of the service as a whole entity. It contains a series of metadata elements that are described through ontologies. Attributes are applicable to each of the metadata elements to define aspects
such as access restrictions, whether a metadata element is mandatory or optional, searchable or not, etc.

Every microservice profile consists of:

- **Global metadata.** This common set of metadata fields (e.g. service name, description, etc.) is pre-defined and their attribute values are fixed. These elements are mainly used by the search engine.

- **Specific metadata.** This optional set of specific metadata can be added and defined by the service creator if needed.

It should be noted that because of their different purpose (e.g. operational, local, searchable – see Figure 2) metadata elements are stored in different parts of the architecture (Figure 3). Listing 1 shows a sample from a typical microservice profile:

```xml
<ServiceProfile>
  <GlobalMetadata>
    <Name>Friend Locator</Name>
    <Description>Sharing your Location with your Friends</Description>
    <Tags>friend, find, map</Tags>
    <ServiceCreator id="servicecreator">mciudad@mciudad-fp7.org</ServiceCreator>
    <Expires>2099-09-17</Expires>
    <URI>/uri">http://myaddress/fl</URI>
    <MaximumConnections>10</MaximumConnections>
    <AccessRights>
      <Grant group="mypeergroup1" permission="execute"/>
      <Grant group="public" permission="create"/>
    </AccessRights>
    <ProviderRequiredCapability>location</ProviderRequiredCapability>
    <Logo>/logo">http://myaddress/fl/logo</Logo>
  </GlobalMetadata>
  <SpecificMetadata>
  </SpecificMetadata>
</ServiceProfile>
```

Listing 1. Microservice profile example.

3.2 **UDL-CD: microservice content description**

This part of the UDL describes the content that a microservice produces and receives by mapping the service content items to a commonly agreed content ontology. The concepts represented in UDL-CD are referenced in the service profile (UDL-SP). The UDL-CD is an ontology offering a broad and extensible set of basic concepts that can be used to characterize the content provide by a microservice. Of course, this is an essential requirement for an effective content-based search.

The UDL-CD is based on common ontologies such as DBpedia (structured data from Wikipedia, a core part of the Linked Data Project – http://linkeddata.org), CIDOC (specialized on cultural knowledge) and TAP (extracting knowledge from the web).
A top-level view of the UDL-CD schema is shown in Figure 4. Listing 2 presents a code snippet of a sample UDL-CD instance.

UDL-CD is an extensible ontology, which means that the KWH administrator has the capability to enrich the set of concepts that are available to the end-user for describing their microservices and their provided content. Ontology matching mechanisms allow to keep this set of concepts minimal. The KWH also employs ontology mapping techniques in order to capture the “implicit semantics” expressed by the end-user when tagging the microservices and their content and to map it to existing concepts. In case no existing concepts are found, a “shallow” ontology will be created.
until new concepts can be mapped to existing ones. Ontology mapping and matching mechanisms are periodically used within the KWH as “routine” management tasks:

```
<Location rdf:ID=“Vienna_International_Centre”>
  <hasLink rdf:datatype = “www.w3.org/2001/XMLSchema#anyURI”>
    http://dbpedia.org/page/Vienna_International_Centre</hasLink>
</Location>
```

Listing 2. Microservice content description example.

### 3.3 UDL-CP: microservice capability profile

As already mentioned before, microservices are generally built out of SSEs. In order to allow a quick, user-friendly, and meaningful service creation (e.g. by recommending only suitable and matching service components) these SSEs are semantically described (Davies, 2009). As the SSEs are different in nature, the way they are described may vary:

- **Embedded capabilities.** These capabilities are part of the execution environment installed in the mobile terminal and are therefore available to all terminals.

- **External capabilities.** This kind of capabilities is linked to functionalities that reside within the network. They are available through the SWH and are discovered by searchable metadata (stored at the KWH). These “external” components can be downloaded and installed in the mobile terminal. After installation they can be used and combined like any other embedded capability.

- **External terminal capabilities.** These are additional capabilities that are provided by terminal makers and telecommunication operators. They are designed to work within a terminal, exploiting their embedded devices.

- **General purpose blocks.** These blocks are not terminal dependent, and could be provided by any third party.

- **Information/content blocks.** These represent pieces of content that exist as such and do not result from the call of a capability.

All of these “objects” exist in the user environment and are manipulated by a user within the SCK. Any resulting microservice will be a combination of such objects with a “touch of control” (conditionals, loops, etc. – Section 3.4). Listing 3 shows the description of a calendar capability for illustration:

```
<EmbeddedCapability rdf:ID = “CalendarCapability”>
  <implementsConcept rdf:resource = “#RescheduleEvent”/>
  <implementsConcept>
    <FunctionalConcept rdf:ID=“AddCalendarEvent”>
      <outputs rdf:resource = “#Success”/>
      <takeInputs rdf:resource=“#CalendarEvent”/>
    </FunctionalConcept>
  </implementsConcept>
  <implementsConcept>
    <FunctionalConcept rdf:ID=“GetCalendarEvents”>
      <outputs rdf:resource = “#CalendarEventList”/>
      <takeInputs rdf:resource=“#Search_term”/>
    </FunctionalConcept>
  </implementsConcept>
</EmbeddedCapability>
```
3.4 UDL-SL: microservice logic
The UDL-SL is a domain-specific XML-based description language that defines the functional and operational aspects of a microservice seeking for high expressiveness and high usability for end-users. However, since high expressiveness unfortunately often results in high complexity, UDL-SL seeks for a tradeoff between both in order to make microservice creation attractive for end-users.

The proposed paradigm mimics a tree structure using a top-down path. Whether for logic or visual layout (Section 3.5), a graphical representation can be achieved by representing nodes within the tree as visual blocks within the SCK interface (which will be explained in more detail in Section 4). Figure 5 shows an overview on how the information flows within a block-based approach.

The logic of a microservice is specified in a UDL-SL file that implements several programming constructs, following a set of programming rules. Typically, this file has two parts: client logic and server logic. Listing 4 shows an example of the structure of a UDL-SL file:

```xml
<Logic xmlns:xsi="..."../schema/mciudad-udl-sl.xsd">
  <ClientLogic>
    <Variable.../>
    ...
    <Procedure.../>
  ...
  </ClientLogic>
  <ServerLogic>
    <Variable.../>
    ...
    <Procedure.../>
  ...
</ServerLogic>
```

Figure 5.
Block-based paradigm of UDL-SL and UDL-SR
<While condition="true">
  ...
  <Wait time = "500"/>
</While>
  ...
</ServerLogic>
</Logic>

Listing 4. Microservice logic example.

The server side program usually consists of an infinite while with at least one wait element inside (for event processing) and a number of variable and procedure declarations.

The client side program may have some initialization code but usually has only variable and procedure declarations that will be used by the UDL-SR components of the user interface or called when some component event is triggered.

In general, the UDL-SL language is specified by:

- a set of data types – basic data types (e.g. text, number, boolean, or date), special data types (e.g. URL, picture, audio, or video), structures and collections;
- a set of operators – comparison, arithmetic, logical and string operators;
- a set of programming rules – syntax and coding styles, program structure, scope of variables and parameters, procedure invocations, syntax of expressions and conditions, event handling and messaging (e.g. for content sharing between providers of the same microservice);
- a set of implicit procedures – for implementing common functionalities (e.g. for managing elements in a collection);
- a mechanism for capability invocation – for invoking both internal (e.g. take a photo) and external capabilities (e.g. read the heart rate from a sensor via bluetooth or display a map from Google);
- a mechanism for remote procedure invocations, e.g. for invoking remote capabilities; and
- a mechanism for system event handling.

The full language specification can be found in Carrez et al. (2009).

3.5 UDL-SR: microservice rendering

The UDL-SR specifies the microservice rendering by defining the set of available graphical elements for the service GUI. The UI elements are independent of mobile device peculiarities, such as varying screen size, limited interaction modalities, and different operating systems. Therefore, UDL-SR defines an abstract set of UI elements to achieve a platform and device independent design.

UI elements defined within UDL-SR are represented through a block-based approach (based on XML similar to UDL-SL, see above). For instance, UDL-SR defines UI elements, such as: presentation, windows, panels, menus, and buttons. These elements are represented in a tree with presentation as the root node and windows as child nodes. Each window also has a child node, namely a panel that can contain additional UI elements. This kind of hierarchy defines the way the UI is rendered.
Listing 5 shows how a DataGrid is defined in UDL-SR as an example:

```xml
<Presentation startingWindow="main">
    
    <Window id = "main" title = "Main window">
        <Menu>
            ...
        </Menu>
    </Window>

    ...
</Menu>

    ...
</Window>

    ...
</window content...>
    <StatusBar text = "any text"/>
</Window>

    <Window id="detail" title = "Detail window">
        <Button text = "Open detail window"
            onClick = "openDetail"/>
            ...
        </Window>
    ...
</Presentation>
```

Listing 5. Microservice rendering example.

### 3.6 Microservice execution

During microservice creation a UDL-SL file and a UDL-SR file is generated (see previous sections). Together, the UDL-SL and UDL-SR files, make up the “executable view” of a microservice and formally define what a microservice is made of, in terms of operational and visual components that are executed by the SEE. Thus, microservices execution on mobile devices is clearly driven by these two UDL files.

Because microservices are fully executed on mobile devices, both the client part (microservice front-end) and the server part (microservice back-end) have to be specified. The UDL-SL defines the logic of the service, both for the client side and for the server side, whereas the UDL-SR defines the elements that form the visual layout of the service, i.e. the GUI for the client. Therefore, overall service rendering at the client side is built with the logic of the client defined in the UDL-SL, merged with the presentation description, given in the UDL-SR. The logic of the server, included in the UDL-SL, defines the service behavior on the provider’s mobile device, such as: when and how content is generated, when and how content is distributed to other providers or served to consumers, how often a specific event has to be triggered, etc.

As already mentioned, the UDL-SL and UDL-SR languages are intended to be platform independent. Microservices are defined in an abstract way for execution on any mobile platform. For the actual execution there are two main options:

1. The SEE includes an interpreter of the abstract definition that is able to process the UDL code, dynamically execute the logic of the server, and build the GUI for the client.

2. The UDL code is pre-processed before being deployed in the SEE so an executable version of the microservice is actually run by the SEE.

While the interpreter-based solution does not need a pre-processing phase, it is certainly more resource-intensive and can lead to performance problems. On the other hand,
the translation-based approach has the advantage that it provides a platform
independent solution for the client side (i.e. just a browser is required for microservice
consumption, which can be considered quite valuable considering the current mobile
platform fragmentation). As a consequence we have chosen UDL translation over
interpretation, designed as follows:

- The UDL-SR, plus the logic of the client in the UDL-SL is translated into
  xHTML+JavaScript code, so it can be executed on a standard mobile browser.
- The logic of the server is translated into pure JavaScript code and run on an
  ad hoc engine embedded into the SEE.

4. Mobile authoring
With the number of smartphone sales surpassing the number of PC sales by the end of
2010 (Weintraub, 2011) and mobile internet usage growing substantially faster than
desktop access we believe in the idea of enabling users to create new services wherever
and whenever they want or need to.

For that matter, we have deliberately chosen service creation on the mobile device over a
desktop approach (like in Google App Inventor for the Android platform – http://
appinventor.googlelabs.com/about/). This section introduces a tool for mobile authoring of
microservices on the mobile terminal, which we have already introduced as the SCK. In the
next section we discuss the challenges and constraints for developing a mobile authoring
tool in a touch-screen based terminal. Afterwards we describe the interaction metaphors
used for creation and edition of microservices. Lastly, we present our SCK implementation
that is based on three user modes to accommodate different levels of user expertise.

4.1 Device constraints
During the development of our authoring tool we had to consider the inherent
constraints of mobile devices, namely reduced screen-size, interaction metaphors,
battery consumption, computing power, cross-platform ability, and storage. In case of
the presented SCK we have selected the Nokia N97 smartphone.

The study of the possible user interface metaphors was consequently limited by the
capabilities available in the Nokia N97 SDK for Symbian OS and ActionScript 2.0 supported
by the Nokia N97 Flash Player. Nevertheless, we believe that the metaphors explored can
also be ported to other current platforms, namely iOS or Android. Furthermore, we think
that our SCK highlights the cross-platform ability of UDL (Section 2.2) and might foster
research on additional platforms. Despite the visualization and interaction metaphors that
can be used across multiple platforms, microservices can be created, updated and deployed
by any tool that is capable of the UDL language. Thus, a single microservice can be shared
between different platforms seamlessly. In the following section, we discuss the interaction
metaphors we have followed while designing the SCK.

4.2 Interaction metaphors
Mobile devices are often used in dynamic, noisy environments and users may be
moving. This makes designing interaction techniques for mobile devices challenging
and classical approaches used on desktops may not always be appropriate
(MacKay et al., 2005). In our case study we have developed the SCK taking advantage
of the available input capabilities of the Nokia N97, namely a touch-screen interface,
where text can be entered through the usage of a virtual keyboard on the screen.
or through the usage of a full physical “QWERTY” keyboard that slides out from the back of the device.

First, we will discuss target selection techniques and afterwards the navigation techniques we have implemented. The evaluation of both techniques was performed through two different evaluation paradigms, namely a “quick and dirty” approach for early feedback and a more thorough and methodical usability test (Preece et al., 2002). We used the “quick and dirty” paradigm to observe and ask users opinions about the specific interface details in an informal approach. We have asked expert evaluators when critique about the techniques was required or colleagues with no prior knowledge in end-user development to find possible misinterpretations of the used techniques. In a more thorough approach, we used a usability test with interviews and an online survey to evaluate the intermediate prototypes. Results can be found in Danado et al. (2010).

In our implementation we have used the following target selection techniques: “mask with available options”, “drag and drop”, “carousel”, and a “dynamic list” (Figure 6). For evaluation of the first technique, mask with available options, we used a “quick and dirty” approach. We have observed that having the previous screen dimmed in the background with the currently available operations on top effectively helped users to get information about the context of the current operation. While selecting an object to perform an operation, users commented that it helped them, as they were able to still access the context of the operation.

The drag and drop technique is a familiar technique used in desktop environments. We used this technique to drag macro-blocks for creation of a microservice.

Based on Poupyrev’s findings that tactile feedback is most useful when used to enhance gestural interaction (Poupyrev et al., 2004), we have added this form of feedback while moving objects with drag and drop. In our “quick and dirty” evaluation, the operation was intuitive for users as they were well used to it from desktop environments.

Another technique used was the carousel. We used the carousel as an intuitive technique for option selection or to traverse through a set of available keywords or tools. When evaluated with Nielsen’s heuristics, a carousel provides control to the user. As an example the user is able to provide input to the system and see his/her option reflected in the system. The carousel provides a consistent approach when used correctly, i.e. the metaphor is consistent while using the UI control; it prevents users from making mistakes, i.e. the system only provides adequate options not allowing the user to select options that may corrupt the mobile service; and, finally, is visually appealing, i.e. the UI control provides a friendly way to present information (Danado et al., 2010).

Figure 6.
Authoring – mask available options, drag and drop, carousel, and a dynamic list.
Finally, we included dynamic lists as a technique for selecting a microservice. This technique is also used in the iPhone and Android devices with success. In our “quick and dirty” evaluation users were also able to use that feature and were also enjoying the physical behavior that made the list scroll according to the finger speed motion.

With regard to navigation techniques, we mainly use buttons to guide the user through the process of creating a microservice. We use buttons as: soft-buttons, available through physical soft-keys on the mobile terminal, or on the screen interface, requiring the screen to be tapped. Accordingly, navigation buttons were designed to accommodate thumb size and to provide an easy selection of the target without frustrating the user. Nevertheless, we also found that navigation only supported by buttons presented one limitation as users were not able to assess the progress of the current task. Consequently, after our “quick and dirty” evaluation, we added cues in the interface to display the task progress.

We have added such cues between the buttons of the service creation wizard for backward and forward navigation (Section 4.3). The “quick and dirty” evaluation showed that such an approach was providing the expected feedback for users but also forced users to move through all questions in order to finish the task. Other approaches can be more successful if users are not forced to go through all the questions like an accordion with a menu expanding or contracting to show additional content or menu options (Kemper et al., 2006). In our case, we have decided to keep the button navigation to shorten development time.

We also introduced a listbox that allows users to easily navigate between windows while creating a microservice (Section 4.3). This approach enables an always accessible metaphor to navigate between windows. The limitation we found was the reduction of available space for creation of the microservice. Our solution to this problem was to allow the user to tap-and-drag (up and down) in a window to move across macro-blocks possibly hidden by the lack of screen space. In the following section, we discuss the implementation of the three user levels and the used interface metaphors.

4.3 Implementation

Being an XML-based language the UDL can be easily represented in a tree structure on a mobile terminal. However, display size restrictions on the one hand and information overload on the other hand make a complete visualization unfeasible. Therefore, we have introduced different levels of complexity where the user can choose from following a “gentle slope” approach (MacLean et al., 1990). The first view aimed at “beginners” is a wizard-like approach asking the user different questions, i.e. the type of the service, or the name (Figure 7 – left). The second, intermediate view uses a drag-and-drop approach for service creation (Figure 7 – middle). The third view targets advanced users with a visual UDL editor (Figure 7 – right).

At any level of expertise users are able to start creation of a microservice from scratch or based on a previous microservice that can be downloaded from the m:Ciudad framework. In order to edit a previously created microservice, a search operation has to be performed through the SRE (Section 2). The search operation can take place as a free text search or through keywords suggested by the SRE. Additionally, the SRE is supported by the KWH, where keywords are semantically related through an ontology. This allows the SRE to further suggest additional keywords from the ontology related to the ones currently selected by the user.
4.3.1 Beginner mode. In the beginner mode users are able to configure simple parameters of a microservice through a wizard-based questionnaire. In this mode a set of questions is presented to the user to customize a microservice. A microservice includes special tags in the UDL-SP (Section 3.1), specifically `<MetadataElement>`, that lets the SCK render the information as questions to the user (Listing 6). Questions can be free text or an option list:

```xml
<MetadataElement Name="sck_1_3_text" Type=""" Value="Foreground Color ">
  <MetadataElement Name="sck_1_3_value" Type=""" Value="#404040"/>
</MetadataElement>
```


During service execution the selected values are accessed as variables within UDL-SR or UDL-SL, see Listing 7:

```xml
<Label text="{repeaterEntry.date}" foreground="{SERVICE_PROFILE.specificMetadata.sck_1_3_value}="/>
```

Listing 7. Code example 2.

4.3.2 Intermediate mode. The intermediate mode is targeted at users that would like to further specify the functionalities of their microservices without requiring them to learn about UDL details. A first version was developed and evaluated extending the wizard-based questionnaire in the beginner mode (Danado et al., 2010). However, we found limitations related to the lack of customization actions when compared with the beginner mode. Thus, we performed “quick and dirty” evaluations with image-based mockups to design a new interface for the intermediate mode. With inspiration from Yahoo! Pipes (http://pipes.yahoo.com) and Microsoft Popfly (www.popfly.com) we selected a drag-and-drop approach.

The intermediate mode follows the concept of connecting macro-blocks, which are one or more SSEs. A macro-block is a component with its own UDL-SR and UDL-SL code and rules that specify how it can be connected to other macro-blocks. These rules were put into an additional file allowing the microservice to be edited in the future in the same mode.
In the intermediate mode’s UI the user can compose a microservice by dragging one or more macro-blocks to the canvas and, afterwards, by connecting and configuring them. Although users had to receive some instructions before being able to fully connect and configure macro-blocks, we found that our approach was intuitive enough for end-users (it should be noted that we have only implemented a subset of possible connections for the prototype for the assessment).

4.3.3 Advanced mode. The advanced mode allows the mobile user to deeply dive into UDL, i.e. service profile, logic, and presentation in so-called “contexts”. Each context provides a “world view”, where users can seamlessly access the nodes of the blocks in that context. The user is able to add, modify, or remove blocks. Recommendations to the user (which are currently based on the inputs a parent node accepts from child nodes) should help in selecting a fitting block.

Figure 8 shows screenshots of the world view of the presentation and a small snippet of UDL that is generated from the SCK. As already mentioned in Section 3.6, UDL acts as an intermediate language – services defined in UDL can be shared among other users and customized or re-edited through the SCK. For direct execution on the Nokia N97 UDL is finally converted into xHTML + Javascript (so it can run in a mobile web server/browser).

At any level of expertise users are able to start creation of a microservice from scratch or edit a previously created microservice. Nevertheless, limitations in the current SCK restrict seamless edition of the same microservice across different user levels. Thus, a service created in the beginner mode can be edited in the beginner and advanced mode but not in the intermediate mode, unless it also complies with the rules in the intermediate mode. Similarly, a microservice created in the intermediate mode can be edited in the intermediate mode and advanced mode. Once a microservice created in beginner or intermediate mode is edited in the advanced mode, and if that service does not comply with the rules stated before, that microservice can only be further edited in the advanced mode. Despite these limitations in the current implementation, we see our work as a significant contribution towards end-user development of mobile services. Furthermore, the UDL proved to support three levels of user expertise seamlessly. Thus, a beginner, intermediate, or advanced user can further customize a microservice with an interface that meets his/her level of expertise.

5. Intermediate evaluation
In order to get early feedback on the authoring tool, the general approach for end-user mobile service creation, and the expectations and background of users, we conducted a
small hands-on user trial and used an online questionnaire in the middle of the implementation process.

5.1 Hands-on trial
All parts of the microservice description language (Section 2.2), such as logics, profiles and content representations, have an impact on the end-user experience. The microservice language part with the closest connection to the end-user is however the presentation part. For the acceptance of the described microservice language solution as a whole, it is crucial to ensure an effective user interaction with the presentation layer of the microservice description language and the associated graphical user interfaces. We have used the outcome of the intermediate user trial to refine our microservice authoring approach.

The intermediate hands-on trial was done with five users (which were rather technologically inclined) performing six different tasks necessary for microservice creation in the beginner and advanced service creation modes. In general, the test showed that users with a technological background can understand the concepts behind the creation of a microservice. For the beginner view the participants recommended usage of navigation tips and less technical jargon to make the process easier. Although the basic concepts of the advanced view were easily perceived, users found it still too complex to use. The tests showed that the interface could be improved through introduction of help tips, short descriptive text, icons, color codes and templates, and, most notably, adopting a “gentle slope approach” to learning (MacLean et al., 1990). This feedback motivated us to design a new “intermediate” microservice creation mode (see Section 4).

5.2 Online survey
The online survey contained 38 questions and allowed to assess the beginner and advanced creation modes of the authoring tool while gathering demographic data, information about the user background, and experiences with similar tools. We received a total number of 52 full responses over a period of 4 months in 2009. Of the respondents 74 per cent were male and 98 per cent from Europe. About 88 per cent had a university degree (bachelor or higher), and 70 per cent were working in ICT or science fields. Of the participants, 44 per cent were between 20 and 29, one-third between 30 and 39, and the remainder (24 per cent) between 40 and 60 years old. Of the respondents, 46 per cent confirmed to access the internet from their mobiles. Two thirds of the respondents use the internet at least 4 hours per day.

The purposes to create microservices were mostly to exchange information (71 per cent), stay in contact (6 per cent), and just for fun (60 per cent) – Figure 9. People would mostly create them while waiting (73 per cent), commuting (61 per cent) or at home (60 per cent). About 44 per cent would also create services at work, i.e. professionally. Free of charge provision/usage was the most frequent answer (73 and 87 per cent, respectively). About a quarter of people could imagine to either make small payments or subscriptions for services.

During the survey users were presented a demo of the question-based SCK (beginner mode) and two videos of both the question-based and the block-based (advanced) service creation approaches. In terms of easiness the question-based approach was received much better (“It seems to be also suitable for beginners to create services”, “The user is well guided by questions”) than the block-based, advanced approach (“for freaks/nerds mainly”, 13 users regarded this method as too complex). The block-based approach was mainly regarded as a professional tool for creating more complex services, and one,
which might fit better on a desktop than on a mobile. Six users, however, mentioned that
the question-based approach might be still too restricting (I could imagine creating
a service this way, but it would probably be too simple to do anything I really wanted.).
This feedback further strengthened our design choice to offer different complexity levels
(see Section 4) in a “gentle slope approach” (MacLean et al., 1990). By this way end-users
can start interacting with the system at a very basic level and during time progress
towards becoming a “power-user”.

6. Final evaluation
The purpose of the final integrated system evaluation was to test the platform with
regard to usability and user experience. Besides, we wanted to get an insight on
real-life situations and areas, where such services would be mostly used, and to get
information on what kind of services the users might want to create with such a
system. The basic usability measures studied during the evaluations were:

• Effectiveness, i.e. the extent to which the intended goals of use are achieved and
  how the framework satisfies the requirements of creating, searching, executing,
  and publishing microservices.

• Efficiency, i.e. the resources, such as time or mental effort, spent to achieve the
goals. Efficiency is characterizing the easiness with which the user is able to
manipulate the system (e.g. short chains and logical order of commands). Therefore,
efficiency is very much related to UI structure, e.g. menus and buttons,
and how fluent they can be used in the system.

• User satisfaction, i.e. the overall satisfaction with the system, how the users like
it and how pleasant the system is perceived during usage. It measures the extent
to which the user finds the system acceptable and enjoyable.

• Value, i.e. the actual importance and value of the system in increasing users’
opportunities to provide and consume contents by using microservices.
6.1 Test setup

The end-user evaluations took place in December 2010 at five different organizations. We have made effort to equalize the evaluation environments and procedures at each test site as much as possible. The actual tests were performed in laboratory conditions.

The m:Ciudad system requirements were used as a reference in the final user tests. For this purpose the requirements were embedded in three storylines, based on the two scenarios outlined in Section 1, which were “walked through” by the test users during the evaluation. The storylines covered the three UI modes developed for beginners, intermediate users and advanced users, and described how the system can be used for searching, creating, publishing, executing, and managing microservices. Successful walkthroughs of the storylines gave evidence that the requirements could be fulfilled satisfactorily.

The evaluations were interactive sessions between the test users and the mentors. During the step-by-step walkthrough of each storyline users were thinking aloud and the mentors could observe the activities of users carrying out the evaluation tasks. The devices used in the evaluations were Nokia N97 (Symbian S60 OS) mobile phones.

After completing the walkthrough of a particular UI mode the user had to answer to a set of questions related to effectiveness, efficiency, and user satisfaction. These questions could be answered on a scale ranging from 1 – “totally disagree” to 5 – “fully agree”. After finishing all the storyline walkthroughs the users were asked to rank their preferences regarding the different UI modes (beginner mode/intermediate mode/advanced mode). Finally, users were presented a questionnaire to assess their individual motivation and situations for creation and usage of microservices as well as the perceived overall value of a service creation infrastructure.

The test users consisted of 13 males and 11 females. The youngest participator was under 20 years old, the age group 20-29 had 15/24 participators (62.5 per cent) while the age groups of 30-39 and 40-49 had 4/24 participators together (16.7 per cent). We think that the age distribution closely matches the prospective audience of the m:Ciudad system.

The main educational background groups of the users were a high school degree (50.0 per cent, 12/24), a bachelor’s degree (16.7 per cent, 4/24), and a master’s degree (20.8 per cent, 5/24). Our test users consisted of university students (54.2 per cent, 13/24), engineers (8.3 per cent, 2/24), researchers (12.5 per cent, 3/24), and other occupations (20.8 per cent, 5/24).

The main working sectors of the test users were the ICT sector (20.8 per cent, 5/24), health and social work (16.7 per cent, 4/24), science and technology (16.7 per cent, 4/24), finance (8.3 per cent, 2/24), other service activities (12.5 per cent, 3/24), and other working sectors (8.3 per cent, 2/24).

Of the users, 16/24 (66.7 per cent) declared that they access the internet from their mobile phone, while eight persons did not. Amongst the mobile internet users, 10 persons were using it less than one hour per day, just one person was using mobile internet 4-6 hours per day. Unsurprisingly, the users were mostly active desktop internet users. Half of them (50.0 per cent) were using the internet for 1-3 hours a day, 41.6 per cent (10/24) used it for 4-6 hours a day and one (4.0 per cent) used it for 6-9 hours a day.

Among the test users, the most actively used technologies were related to social networks (such as Facebook, which 21/24, 87.5 per cent of users were using) and chats (17/24, 70.8 per cent). Additional technologies of interest were bulletin boards and forums (12/24, 50.0 per cent), podcasts (10/24, 41.7 per cent), web languages (e.g. HTML,
8/24, 33.3 per cent), RSS feeds (8/24, 33.3 per cent), mobile services (7/24, 29.2 per cent), blogs (7/24, 29.2 per cent), mashup editors (2/24, 8.3 per cent), semantic web (2/24, 8.3 per cent), and web services (7/24, 29.2 per cent).

6.2 Results
In the user evaluations all of the three UI modes (beginner mode/intermediate mode/advanced mode) were assessed. Besides standard usability testing we were particularly interested in the feasibility of the different UI modes in mobile end-user environments.

As already explained in Section 4 the different user modes were characterized as follows:

(1) Beginner mode (BM):
- The creation of the service is based on the customization of already existing microservices (templates).
- The user can enter the meta information describing the service as well as change some characteristics of the service look and feel.

(2) Intermediate mode (IM):
- The creation of the service is based on dragging visual service elements or blocks to a canvas and creating links between them (e.g. one block is the parent of another one).
- The user can create a service from scratch using high-level components.

(3) Advanced mode (AM):
- The creation of the service is done using basic (low-level) components (both user interface and logical constructs).
- The user can create a service from scratch or tweak some lower level components from an already created service.

6.2.1 Effectiveness. The BM storyline consisted of 18 steps (tasks), which were followed in total by 19 questions related to effectiveness. The mean of all scores was 4.3, and the standard deviation was 1.1.

The IM storyline consisted of five steps of tasks followed by six questions related to effectiveness. The mean value of all scores was 4.5, and the standard deviation is 1.1.

The AM storyline consisted of four steps of tasks followed by four questions related to effectiveness. The mean value of all scores in this mode was 3.3, and the standard deviation was 1.8.

The mean value of the scores in the IM is slightly higher compared to the BM. We suspect that (and also because of user comments) the effectiveness of the IM was perceived higher than in the BM, however statistical significance could not be reached. In contrast, the mean value of the scores of the AM is clearly the lowest. This suggests that this UI mode was not very efficient, because the users found it difficult to get the requested things done in that mode.

Figure 10 shows a percentual distribution of all given scores related to effectiveness in the three different modes. The shape of the distribution (and the mean of all score values) in the BM and IM indicates a good performance and satisfaction towards reaching the goals in these two modes. In contrast, the relatively high columns with
small score values (1 and 2) of the AM suggest that using the system with this UI mode might have been too difficult for some users while trying to reach their goals during the evaluation.

6.2.2 Efficiency. After the walkthrough of each mode’s storyline a set of four efficiency-related questions were presented to the users. The mean values and standard deviations (in parenthesis) of all scores were 3.6 (1.2), 3.8 (1.1) and 2.4 (1.5) in the beginner, intermediate, and advanced modes, respectively.

The mean of all score values in the IM was slightly higher compared to that of the BM (similar to the effectiveness score, see above). On the other hand, the mean value of the scores in the AM was clearly lower compared to that in other modes. This suggests that the AM included too complicated chains of commands, which were difficult to remember, and that using the system in this mode required some technical understanding about programming that were beyond the skills of an average end-user.

Figure 11 shows the percentual distribution of all given scores related to efficiency in the beginner, intermediate, and advanced modes. The distribution of given score values in the BM was slightly more evenly distributed compared to the previous effectiveness distribution presented above. The more frequently given lower scores are highlighting the challenge of constructing an efficient user interface (e.g. including fluent and short chains of commands in logical order).

The distribution of scores in the AM has a high column of “1” clearly showing the complexity of the AM in the users’ opinions, as stated before.

6.2.3 User satisfaction. A set of three questions related to user satisfaction was asked after using the system in each UI mode. The mean values and standard deviations (in parenthesis) of all scores were 3.6 (1.2), 4.2 (0.9) and 3.1 (1.4) in the beginner, intermediate, and advanced modes, respectively.
The mean of score values in the intermediate and beginner modes suggest that the users preferred the IM over the BM.

Figure 12 shows the percentual distribution of all given scores related to user satisfaction in the beginner, intermediate, and advanced modes. The distribution shows that in the intermediate mode the two highest score values constitute about 80 per cent of all given score values indicating a high level of acceptance. In the AM, the distribution shows a relatively even distribution of the score values. Obviously this mode divides the users’ opinions. There are as many of those who like the mode as those who do not like to use it.
6.2.4 Value. Estimation of the value of the m:Ciudad concept was done by asking the users about the actual importance and value of the m:Ciudad system in increasing their opportunities to provide, consume, and distribute contents by using microservices. The four questions related to value were presented to the users after they had finalized their storyline evaluations by using different UI modes.

The summary of the individual questions, their means and standard deviations are presented in Table I. Many users have expressed that the idea of having the possibility of creating and using microservices is fascinating, which can be seen also from the mean value of the first question. On the other hand, the m:Ciudad way to solve the problem leaves some things to be improved – the means of the last three questions are lower than the mean value describing the idea itself.

6.2.5 Ranking the UI modes. The users were asked to rank the beginner, intermediate and advanced modes right after they had finished the storyline evaluation walkthroughs by giving the following score values: “1” (high preference), “2” (medium preference), or “3” (low preference) for each UI mode. The detailed tasks were as follows:

- Rank the modes (in general) regarding how you feel you could reach the goals, which you carried out with different user modes during the tests (related to effectiveness).
- Rank the modes regarding how error free, easy, intuitive and stressless your usage experience was with different user modes (related to efficiency).
- Rank the modes regarding how interesting, meaningful and pleasant was your experience during the manipulation of different modes (related to user satisfaction).

The results showed that the BM was the most acceptable from an effectiveness and efficiency viewpoint. In these categories the AM was the least favorable mode. This differs slightly from the results received in the context of the storyline walkthroughs, where the means of the scored values were highest for the IM. This difference may be due to the fact that during the walkthroughs part of the questions were presented before the user had experienced all the UI modes and the final ranking was made after seeing all the creation modes. Regarding the user satisfaction aspect the IM was mostly preferred (both in the storyline evaluation and in the final ranking exercise), which indicates that it is the most acceptable in users’ opinions.

6.2.6 General user feedback. At the end of the evaluation session the users were asked to clarify:

<table>
<thead>
<tr>
<th>Question</th>
<th>Mean result</th>
<th>SD</th>
</tr>
</thead>
<tbody>
<tr>
<td>How would you rate the value of the Microservice idea at all? (1 = low and not useful, 5 = high and useful)</td>
<td>4.0</td>
<td>0.8</td>
</tr>
<tr>
<td>How would you rate the m:Ciudad way of solving the idea of utilizing Microservices? (1 = poor and clumsy, 5 = innovative and interesting)</td>
<td>3.6</td>
<td>1.2</td>
</tr>
<tr>
<td>What value would you give for usefulness of executing microservices on your mobile? (1 = low and not useful, 5 = high and useful)</td>
<td>3.6</td>
<td>1.1</td>
</tr>
<tr>
<td>What value would you give on possibility to publish microservices made by yourself? (1 = low and not useful, 5 = high and useful)</td>
<td>3.5</td>
<td>1.1</td>
</tr>
</tbody>
</table>

Table I. Questions measuring the value of the microservice idea
the purposes for which they possibly would be interested to create their own microservices; and

- the situations and places, where they possibly would use or create microservices.

Almost 80 per cent of the users would create microservices just for having fun. Also, staying in contact with other people is an important motivator here – 75 per cent of users would use microservices for that purpose. Exchanging information and things might be a motivation for 54 per cent of users. Moreover, creating new business opportunities and meeting people (42 per cent of users) and selling things (33 per cent of users) were seen as possible triggers for microservice creation.

Regarding the places of microservice creation, home was the most natural place for 75 per cent of users. Travelling (67 per cent) and while waiting (58 per cent) would also be frequent situations for creating microservices.

The usability evaluation produced a list of ideas and suggestions how to improve the system. Being a proof of concept, the m:Ciudad platform was able to offer a versatile set of functionalities, though the system still needs more refinements. Most criticism among the users during the evaluation was caused by the difficulty of using three different applications (SCK, SEE and search engine) with different UIs and making the right choices between them. For example, previewing a newly created service needed switching from the SCK to the SEE GUI (i.e. the browser) and then back again to the SCK. The existence of three tools in the mobile device should be hidden from the users’ view. A unique entry point should allow the users to access the different modules seamlessly: searching, creating, and executing. Hence, a relatively small effort and redesign of certain parts of the user interfaces might probably lead to a remarkably better usability of the system.

According to the user evaluations, both the BM and the IM are worth further exploration. Especially the IM seemed to be promising regarding future development of the system. The AM seemed to be too complicated and difficult and it is not acceptable for mobile phones and their users at present, but it might be feasible on a desktop PC instead. It seems to be an appropriate option for making specific modifications to already existing microservices.

7. Related work and advance beyond the state of the art
The work presented here advances the state of the art in three main areas, namely:

1. end-user programming and service creation;
2. semantic web services (SWS); and
3. user-generated content (UGC) and social semantic web.

7.1 End-user programming and service creation
Current examples for user-created software on the web include mashup tools like Yahoo! Pipes (http://pipes.yahoo.com) or Microsoft Popfly (www.popfly.com). In the mobile world, widget platforms such as Nokia WidSets (www.widsets.com) have emerged that, at a first glance, could be seen similar to our approach. However, these offerings do not provide a platform for construction of real services (and are limited to content aggregation not provision), and can be seen complementary to our work as new microservices could be developed on top of them.
The innovations of our solution compared to existing platforms and enablers (Duke et al., 2004; Kuropka and Weske, 2008; Tarkoma et al., 2007; Shin et al., 2008), end-user development techniques (Lieberman et al., 2006; Burnett et al., 2004), and mobile service composition approaches (Kazhamiakin et al., 2008) are:

- a platform-independent and ontology-based service description language for mobile devices;
- a mobile execution environment for user-generated services; and
- new end-user oriented service creation methods that run on the mobile.

7.2 Semantic web services
SWS are typically based on ontological service models such as OWL-S (Martin et al., 2004) or WSMO (de Bruijn et al., 2005) that are extensive, but rather heavy-weight for end-users’ involvement. Although more lightweight approaches such as the semantic WSDL extension SAWSDL (Farrell and Lausen, 2007), SA-REST (Lathem et al., 2007), and Microformats exist, they are not well suited for mobile services as they overlook essential aspects necessary for the deployment on mobile phones, e.g. content rendering specifics. Recent efforts towards modeling of mobile service infrastructures (Villalonga et al., 2007; Kim et al., 2006) might be too generic and not be widely accepted or standardized yet. Moreover, existing toolkits for SWS (Kerrigan and Mocan, 2008) are not oriented towards end-users due to their complex nature. To fill the gap, our work offers a description language for mobile user-generated services and user-friendly ways for mobile service annotation while on the go.

7.3 User-generated content and social semantic web
The presented framework builds on top of the widespread Web 2.0 trend of UGC and extends the concept by allowing users to create their own services. The aspects of end-user perception and acceptance of such framework for the generation of microservices as well as the usage of semantic social software and community-driven practices (Auer et al., 2007) are of large relevance as well. Microservices are a quantum leap over UGC and social web since they:

- are dynamic, changing their output over time and letting users move beyond static content into functionality;
- give users full control over their creations, since they are provided from their own mobile terminals; and
- allow for multiple authoring, since a microservice can have several providers (Carrez et al., 2009).

However, microservices respond to the same ultimate user needs as UGC: authorship, keeping in contact, self-broadcasting; and they have an equivalent viral potential, creating a service metropolis which is a source of relevant, on-the-go information for other users.

8. Future work and conclusions
In this paper, we have presented a novel design for user-generated mobile microservices. An extensible model for the definition of a microservice as well as a mobile authoring tool for end-users have been developed. This sets a step towards the vision
of user-generated mobile microservices and their deployment in an open service platform. A thorough assessment of the final prototype with a substantially large set of users has been performed in the form of lab tests.

The m:Ciudad platform is, to our knowledge, the first designed and implemented infrastructure enabling end-user mobile service creation. Certain aspects of its usage, such as applicability of the derived solution to different mobile platforms and its execution on the users’ mobile devices remain to be investigated. A promising exploitation direction is seen in the context of emerging telco-driven platforms enabling third-party service developers using APIs to telecommunication service infrastructures, such as Wholesale Applications Community (www.wacapps.net). Functionalities for facilitation of efficient service creation, discovery of microservice elements, and associated knowledge management might be enhanced. Such advanced techniques would evolve the current “shallow” Web 2.0 of content to the future “semantically structured” internet of services. The technical characteristics of the resulting system such as its performance in a large-scale evolving internet environment are to be evaluated as well.

The general system and the different levels of complexity in the user interface are aligned with exploitation and business considerations. As seen in the intermediate and final evaluations, the purposes of creating services would be mostly to exchange information (71 per cent), stay in contact (60 per cent), and just for fun (60 per cent) on the basis of non-commercial use. Thus, the exploitation model to approach the customer would be to run such microservice creation platforms for free, offering specific premium features on a per use or subscription basis.

Notes
1. In this work, we represent knowledge as semantically annotated data and content.
2. We are aware of the fact that currently only few mobile operators allow direct network connections between two mobiles though.
3. $U^+$ Service is a project-internal term for microservices.
4. IOPE stands for “Input/Output/Pre-condition/Effect” as in the OWL-S service profile (Martin et al., 2004).

References


About the authors
Marcin Davies (Dipl.-Ing.) holds a Master degree in Computer Science from the Vienna University of Technology, where he also pursues his PhD. Since 2006 he has been a Researcher at the FTW Telecommunications Research Center Vienna. With years of professional experience in software development and design, knowledge and content management, his current research activities focus on the semantic web, mobile services and end-user development. Marcin Davies is the corresponding author and can be contacted at: davies@ftw.at

Dr François Carrez received a PhD in Theoretical Computer Science from the University of Nancy, France in 1991. Since then he has worked for 18 years for the Alcatel Research Centre in Paris, in the areas of Security, Distributed Artificial Intelligence, Ad-Hoc Networking and Semantics. He joined the CCSR forces at the University of Surrey, UK in 2006 and is currently Project Manager of the FP7 Coordination Action “Internet of Things Initiative”.

Juhani Heinali, Lic.Sc. (Tech), became Master of Science in Physics at the University of Turku (Finland) in 1980 and was awarded a Licentiate in Technology degree at Tampere University of Technology (Finland) in 1992. He joined the VTT Medical Engineering Laboratory in 1982. Nowadays he works as a Senior Research Scientist at the Context Awareness and Service Interaction Knowledge Centre at VTT. At present, his primary interest is the research of human-technology interaction (HTI) and human-centered design (HCD) of applications and services related to health and wellbeing. He has worked as a project manager at VTT in several national R&D projects in the field of information technology. Moreover, he has participated in several international projects as a Researcher in European projects related to HTI and HCD.

Dr Anna Fensel is Senior Researcher at FTW Telecommunications Research Center Vienna and a Lecturer at the University of Innsbruck, Austria. She has a Doctoral degree in Informatics from the University of Innsbruck (2006), and a university degree in Mathematics and Computer Science from Novosibirsk State University, Russia (2003). She has been active in semantic research for a decade and authored more than 50 refereed papers in international journals, conference and workshop.
Maribel Narganes is a Senior Researcher in the Service Platforms field at Tecnalia Research and Innovation. She received a BSc in Computer Science from Deusto University (Bilbao, Spain) and an MSc in Telecommunications and Information Systems from Essex University (Colchester, UK). During her tenure with Tecnalia (previously Robotiker), she has participated in numerous design and development projects related to embedded software for telecommunication equipment (GSM, GPRS, ISDN, ATM and SDH/SONET), with special focus on real time architectures. Her primary research experience is in the area of network management technologies, broadband technologies and IP networks. Currently, her research focus involves user-centric mobile services and platforms.

José Carlos dos Santos Danado was an ERCIM post doc at IDI/NTNU from September 2010 until June 2011 and works as a Researcher in the ArTe project (www.artentnu.com). He was awarded a PhD from the University of Evora, Portugal in 2008, in the research field of ubiquitous computing and location systems. In the last two years, he has worked at YDreams (www.ydreams.com/) as the Technical Leader of two EU projects FP7, m:Ciudad and Mugges.