2009

SERVICE-BASED INTERACTIVE WORKFLOWS FOR MOBILE ENVIRONMENTS

Sonja Zaplata
Distributed Systems and Information Systems, Computer Science Department, University of Hamburg

Dirk Bade
Distributed Systems and Information Systems, Computer Science Department, University of Hamburg

Ante Vilenica
Distributed Systems and Information Systems, Computer Science Department, University of Hamburg

SERVICE-BASED INTERACTIVE WORKFLOWS FOR MOBILE ENVIRONMENTS

Sonja Zaplata, Dirk Bade, Ante Vilenica[1]

Abstract

Since the use of mobile devices spreads increasingly, mobile systems also play a major role for distributed business processes. In such scenarios, extending workflow management support to mobile systems offers potential to seamlessly integrate field staff into business processes, even if executing devices are disconnected from the company's server. However, the heterogeneity of current mobile systems still requires complex device-specific descriptions of user interfaces to integrate manual tasks. Therefore, this paper presents an abstract and modality-independent description model to support the development and execution of interactive mobile workflows and a corresponding prototype realization based on a service-oriented execution module.

1. Introduction

In many companies, field staff is often already equipped with several mobile devices, such as notebooks, PDAs, or mobile phones. In order to support company-wide distributed business processes which include these employees, mobile devices shall be integrated seamlessly into such processes − despite of their specific technical differences. In such a scenario, mobile devices could be upgraded to run their own (partial) workflows, if the provided workflow management system can adapt to their specific needs. For such a purpose, specialised process description languages and workflow management systems have emerged which describe and execute the control flow of such applications. Unlike traditional workflow systems which are mostly centralized and include mobile clients as participants of single tasks only, more advanced systems focus on the distributed, self-contained and preferably disconnected execution of business processes on mobile devices by an integrated lightweight workflow engine (e.g. [5, 7, 13]).

For dealing with such mobile workflows, manual tasks containing user interactions play a key role. This is primary due to the fact that users carry mobile devices in order to *interact* with them at any time and place and are using mobile applications which are often related to the user's posisition or situation. Therefore, business processes executed on mobile devices are rather characterized by context-dependent interactive applications than by automated processes running in the background. Additionally, there are situations where certain interaction modalities are prohibited or inappropriate. For instance, using speech during theater performances or in noisy environments like factory work floors is inappropriate, whereas interaction involving eyes and hands is forbidden

1 Distributed Systems and Information Systems, Computer Science Department, University of Hamburg, Vogt-Kölln-Str. 30, 22527 Hamburg, Germany

while driving a vehicle. Thus, a particular challenge for the development of mobile workflows is the specification of interaction tasks and the appropriate integration of manually initiated applications. Due to the heterogeneity of mobile devices, operating systems and network facilities, there is a great variety of possible user interfaces, reaching from audio and video to audio-visual interface modalities. Even within the field of the visual interface modality, screen size, resolution and underlying software systems may vary highly, so interactive applications have to be developed platform-dependent with substantial knowledge about the specific device, its software system configuration [10] and its current situation. Consequently, as a workflow participant, the executing device and its context must be known in advance and each interactive task has to be tailored to each single type of device and for more than one situation, which is in many cases very costly and inflexible. In order to cope with such challenges, an appropriate model for user interactions has to be developed which is independent of a specific platform and context - meaning any combination of mobile device, operating system, and user interface modality. However, rather than developing *n* user interfaces for *n* platforms, we suggest to develop only one *abstract* user interface model which is then automatically tailored to specific platforms at runtime. In addition, such a platform-independent approach has to consider the special characteristics and limitations of current distributed mobile workflow management systems, e.g. the requirement to work disconnected.

In order to develop a systematic approach, the following section presents a use case to motivate and identify important requirements for such a platform-independent workflow interaction model. Subsequently, existing work in this research area is analyzed and evaluated in order to provide a basis for an enhanced approach presented in section 3. In section 4, the feasibility of modality-independent user interactions is shown by the integration of the developed processing module into an existing workflow implementation. The paper concludes with a summary and an outlook on future work.

2. Use Case: Parallel Disconnected Mobile Workflow Processing

A typical mobile workflow scenario can be found in the area of field work for marketing research where a high number of mobile users execute the same workflow in parallel. This section summarizes the experiences, observations and requirements of two German marketing research facilities being interviewed on this subject.

Marketing surveys have previously been performed by paper questionnaires which had to be produced and distributed, filled out manually and finally had to be collected or sent to the research institution, where they had to be typed into the computer in order to be integrated and evaluated. As this procedure is rather inflexible, costly and time-consuming, the integration of mobile devices could eliminate several of these drawbacks. Due to the fact that marketing research is often characterized by casual employment, the assignment of few special purpose devices is not profitable, but - controlled by a central management system - the existing infrastructure of casual employees (e.g. personal mobile phones) can be utilized to transfer and present questionnaires electronically. First attempts towards the utilization of mobile phones involve the bidirectional exchange of SMS (Short Message Service) or the use of browser-based web applications which both cause unacceptable costs because of their need for frequent or permanent network connections.

Another main aspect is the integration of user interactions with automated or semi-automated tasks and application calls. As many interactive applications are only valuable in a specific context, such as performing a measurement at a specific location, the utilization of simple user interfaces, device-specific interactive applications and even fully automated services is highly interrelated. A use case related example is the integration of a GPS application to attach the current position of market researchers during the interview. Therefore, the ability to integrate user interactions with arbitrary (semi-)automated activities such as software services is an important requirement to build interactive workflows for more than one specialised application or business domain. The necessary integration therefore also implicates an open architecture and an adequate data exchange mechanism, in order to allow control flow decisions based on data entered or automatically processed in previous activities.

Figure 1 shows the overall infrastructure for an exemplary survey with electronic questionnaires which are distributed to mobile market researchers: Here, the participating users download the workflow description and start to execute it in parallel, e.g. at different locations. An exemplary workflow is depicted in figure 2. It contains three activities, two of them are user interactions asking the market researcher to enter information about the interview. The last activity calls an application in order to obtain position data of the current location and attaches it to the result set, which is finally returned to the server.

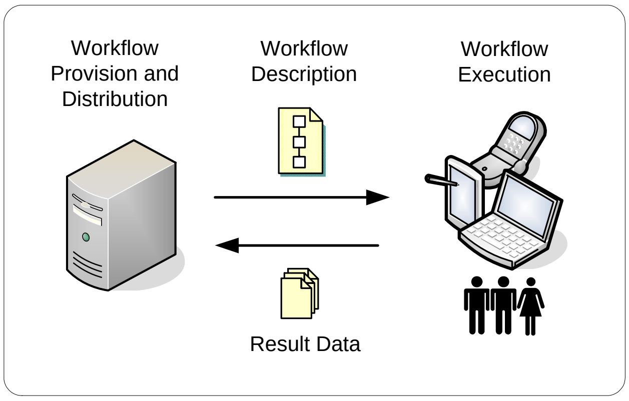

Figure 1: Parallel disconnected mobile workflow processing

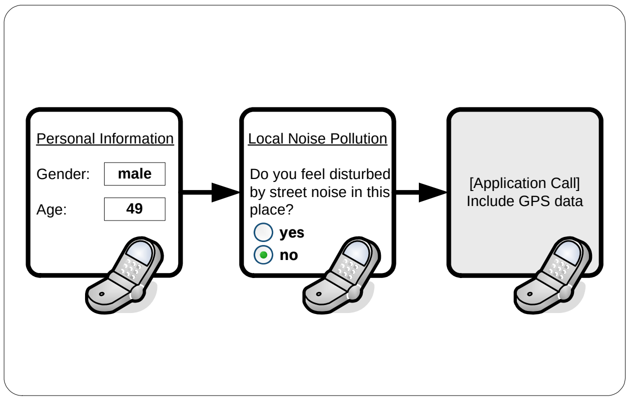
Personal Information
Gender:
male
Age:
49
Local Noise Pollution
Do you feel disturbed by street noise in this place?
yes
no
[Application Call] Include GPS data

Figure 2: Example workflow for electronic surveys

However, with a growing number of field workers, the number of different executing devices increases significantly and the customization of otherwise identical user tasks to each single executing device would be very expensive and, considering the assumed cost-value ratio of such market surveys, not profitable. Therefore, a one-for-all workflow description which is also able to access device-specific applications would be advantageous. Such an approach to mobile workflow management systems leads to increased requirements resulting from the heterogeneity as well as from the limitations of mobile systems (cp. [10, 16]). The next section will therefore analyse and evaluate existing work in the area of workflow related user interface descriptions in order to develop an appropriate solution to this problem.

3. Analysis of Existing and Related Work

There are two main directions for interactive workflow processes executed on mobile devices − *thin client* and *fat client* (resp. *rich client*) solutions. In the first case, mobile devices present information to and receive input from the user, but the business logic resides on a server in the backend. A common approach realising this is to use web applications running on a central application server, rendering information as HTML pages, which are transferred with e.g. HTTP or WAP to a (mobile) device to be displayed using an internet browser. However, this approach has several drawbacks: For example, an online connection has to be kept as long as the user interacts with the application, which is often neither possible (e.g. indoors) nor preferable (e.g. costs, energy consumption). Furthermore, interaction possibilities are constrained to the elements of HTML or WML and activities which need to access resources of a mobile device (e.g. storage, camera, GPS) are not executable, because such resources cannot be accessed by an internet browser. Finally, application development is more difficult as the server is responsible for managing user sessions and dealing with unreliable network connections.

An example for an interactive thin client approach is the *Java Border Service Architecture* [10]: It allows using a single application instance on multiple heterogeneous devices. This is accomplished by taking a snapshot of an application's presentation layer at runtime, transforming it into an abstract intermediate representation before converting it according to the (mobile) device's capabilities. Interaction events (e.g. pressing a button) are exchanged between the application kernel and its presentation layer copy by sending messages over a network link. Other solutions rely upon *Ajax* (Asynchronous JavaScript and XML), which allows to asynchronously fetch data from a server in response to user interactions and to update the HTML interface correspondingly. These approaches, although dealing with some of the aforementioned problems, are not suitable for mobile environments in all cases, as e.g. a stable network link is required but not always available.

On the other hand *fat client* solutions have been developed. In this case, the application as well as the data is stored on the device, user input is processed locally and a network connection is only used to communicate results back to or receive new tasks from servers. Java Micro Edition runtime [17], which is installed on many mobile devices and is most commonly used for developing such applications, does not only allow for arbitrary interaction methods (forms, 2D- and 3D graphics, voice, etc.) but also grants access to most of a device's local resources through standardized APIs. This way, applications may run independently of a network connection, sessions may directly be stored on the local device and once a network connection is available, results may be sent back to a server for further processing. There are several corresponding workflow engines available: The architecture presented by [13] describes a fully service-based workflow engine, running on mobile devices. Users interact with the engine by navigating and communicating through XHTML pages and forms. The results are wrapped in SOAP messages and sent to a local WS-BPEL workflow engine, managing the control flow of the process. This approach focuses on XHTML to describe interactive activities and therefore lacks several interaction possibilities as it is depending on visual modality only. As another example, the *Active Forms* runtime environment [14] addresses the integration of arbitrary applications, accessible through web service wrappers, into an XHTML- or mobile forms-based user interface. User tasks are described using WS-BPEL [12] and executed by a lightweight workflow engine to orchestrate user interactions spanning multiple applications. As WS-BPEL alone is not appropriate for describing user interactions, two language extensions have been defined: BPEL4People [1] and WS-HumanTask [2]. These allow to specify manual tasks within a workflow, but lack the possibility of describing a user interface in detail. Moreover, WS-BPEL itself requires static endpoint references, which is a big drawback if being used in workflows spanning multiple devices in dynamic environments.

Summarized, the presented workflow engines in common either require a steady network connection and/or lack the possibility of processing a rich description language for interactive workflow processes. Such a language is used to describe (interactive) activities by defining interaction components as well as flow elements specifying the temporal-logical course of actions. Existing languages can be classified by the artefacts used to define an interactive workflow and its activities. Two approaches can be distinguished: *abstract* and *model-based descriptions*. The *User Interface Modelling Language* (UIML) [11], for example, is a representative of the abstract description family. UIML allows defining interface elements in an abstract – device independent – way, but lacks a mapping to concrete presentation components as well as the integration of control flow logic. The *Extensible User Interface Language* (XUL) [4] is another abstract description language, which is not – in contrast to UIML – transformed to a specific language, but interpreted at runtime. It is used, for example, by Mozilla's Firefox applications to create the user interface and is interpreted by the Gecko Rendering Engine at runtime. But as XUL can only describe graphical user interfaces, it is bound to the visual modality. Similar drawbacks also hold for other abstract description languages, e.g. MXML [3], XAML [8] and XForms [18].

A popular representative of model-based approaches is the *ConcurTaskTree* (CTT) [15]. *Tasks* a user has to perform are decomposed into smaller subtasks leading to a task tree. The tree's leaves represent the interaction components and are connected by binary operators to state, e.g. whether two tasks have to be executed sequentially or in parallel. Additionally, *dialogs*, *presentations* and *users* are part of CTT's vocabulary and are used to group tasks to further specify the control flow and to personalize interfaces. The drawback of native CTT as a description language has its roots in the orientation of CTT towards client/server environments. This leads to an insufficient support to model data and control flow as well as non-functional requirements. Nevertheless, CTT is most suitable to form a basis of an abstract interface language for mobile workflows, not only taking into account the design of multimodal interaction methods but also the dependencies between different activities and the heterogeneity of platforms. Moreover, a graphical modelling tool called TERESA [9] supports the development of CTT-based interface descriptions.

4. Service-based User Interactions for Mobile Workflows

Workflow-based applications such as presented in the use case (cp. section 2) require a platform-independent interaction model and an appropriate processing module to relieve process designers from considering the system-specific behaviour of each executing device and its potential situation. Based on the experiences of existing approaches as analyzed in section 3, this section first introduces an *abstract interface model* to describe user interactions in a platform-independent way. In the following, a prototypical realization of a corresponding *mobile interaction service* as a loosely coupled workflow management module is described and integrated into an existing workflow management system. Finally, experiences with the technical evaluation of the developed prototype are pointed out.

4.1. Abstract Interface Model

The *ConcurTaskTree* (CTT) [15] has proved to be a suitable approach to form the basis of an abstract interface model for mobile workflows, but still lacks an appropriate description of data and control flow. The developed enhanced model is therefore made up of three main artefacts: One represents elements to define the user interface, one for the control flow and another for the data flow. Additionally, there are elements to define non-functional requirements as well as stylesheets. Due to space limitations, this section focuses on the artefact for the user interface and gives only a brief overview of the other artefacts as they correspond to traditional abstract workflow languages such as described by the Workflow Management Coalition (WfMC) (cp.[19]).

From a high level point of view, user interactions can be characterized as a sequence of manual tasks of data input and output. To describe elements dealing with user interaction in a platform-independent way, the proposed model includes a preceding interface layer. Key concept within this layer is a meta-description of user interface elements. At runtime, the abstract user interface model, e.g. the elements of the interface layer, is automatically transformed into a platform-dependent description. This mapping takes care of the specific capabilities of each platform.

For the development of platform-independent user interface models, basic user interface elements as needed in interactive business processes have been identified. Figure 3 shows the resulting structure of the interface model, containing the most relevant elements and attributes. Centrally, there is a need for elements covering the input and output of data edited by the user. This group of elements is called *Edit*. Second, there is a demand for elements which deal with data that cannot be edited but selected - referred to as *Selection*. Third, elements are needed to trigger actions or to navigate within user interfaces which are called *Control*. In order to model abstract user interfaces, any of these three elements can be combined arbitrarily. Thus, an abstract user interface model, called *Presentation*, consists of several of those basic user interface elements. A full mobile workflow or *Process*, can consist of several *Presentations*. It is important to mention that the three basic user interface elements can be tailored towards a special usage by adding certain meta-descriptions. For example, *Selection* can either be *Single-Selection* or *Multiple-Selection* indicating if one or more elements can be selected from a set of elements, e.g. a list. Moreover, *Edit* elements can predetermine a certain data type, such as *String* or *Decimal*, to facilitate further processing.

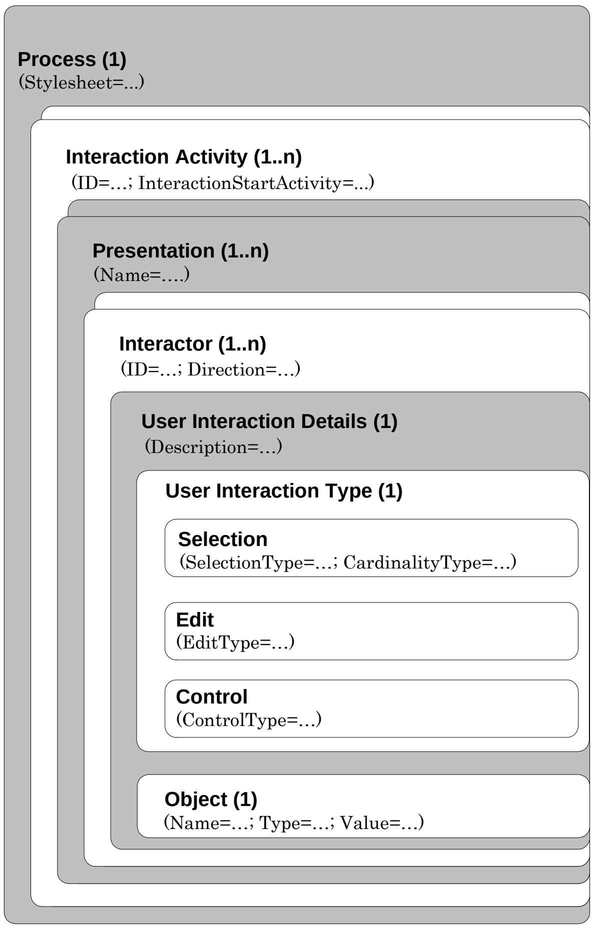
Process (1)
(Stylesheet=...)
Interaction Activity (1..n)
(ID=...; InteractionStartActivity=...)
Presentation (1..n)
(Name=....)
Interactor (1..n)
(ID=...; Direction=...)
User Interaction Details (1)
(Description=...)
User Interaction Type (1)
Selection
(SelectionType=...; CardinalityType=...)
Edit
(EditType=...)
Control
(ControlType=...)
Object (1)
(Name=...; Type=...; Value=...)

Figure 3: General structure of the abstract user interface model (selected attributes only)

```
01 Process Name="Noise Pollution Survey"

02 InteractionActivity ID="First"
       InteractionStartActivity="True"

03 Presentation Name="ZipCode"

04 Interactor ID="Zip" Direction="Out"
05 UserInteractionDetails
       Description="Enter your Zip Code:"
06 UserInteractionType EditType="String"
07 Object Name="Code" Type="String[]" Value=""

08 Presentation Name="Street/Opinion"

09 Interactor ID="Street" Direction="InOut"
10 UserInteractionDetails
       Description="Select your street in $Zip:"
11 UserInteractionType
       SelectionType="SingleSelection"
       CardinalityType="High"
12 Object Name="Streetlist" Type="String"
       Reference="Remote"
       Value="http://www.streetlists.org/$Zip.xml"

13 Interactor ID="Opinion" Direction="Out"
14 UserInteractionDetails
       Description="Noise Pollution:"
       Optional="True"
15 UserInteractionType
       SelectionType="SingleSelection"
       CardinalityType="Low"
16 Object Name="Choices" Type="String[]"
       Value="[Low|Medium|High]"
```

Listing 1: Exemplary interaction activity from the market research domain

There are various options to turn an abstract element into a concrete one. Listing 1 gives an example of two user interface descriptions (lines 3-7 and 8-16) using the introduced model. As mentioned previously, the platform-independent model also has artefacts to describe the control and data flow. Therefore, each basic user interface element (called Interactor) has a data output and input container to which user input is written or can be read from respectively. An example can be found in lines 9-12, where $Zip is as well input as output parameter (indicated by Direction="InOut") and refers to the contents of the Interactor with the respective ID in line 4.

Furthermore, these data containers support the use of references which allows the integration of local or remote elements within a user interface. Accordingly, in the given example contents of a selection element can be downloaded at runtime (line 12), which is advantageous if the actual execution of an activity is optional or is, e.g. to be decided in dependence of previous input data. Provided that a network connection can be established, this option allows to save memory space and to load large objects only if they are actually required. Finally, the proposed model gives the opportunity to specify non-functional requirements and stylesheets (not depicted). With these two groups of elements it is possible to bind a user interface element to a special interaction modality (for example, displaying a large image might need a certain screen size) or to specify the Look-and-Feel of the application. By ignoring these additional descriptions, the respective interpreter can initialize a fallback to other (suboptimal) modalities if applicable.

4.2. Mobile Interaction Service

The developed mobile interaction service is a device-specific component which can be attached to the mobile system's workflow engine or can be utilized as a lightweight stand-alone system running on a mobile device. It hides platform-specific characteristics from the process designer such that it does not matter which mobile device is used or which interaction modalities is supported. Therefore, its main tasks are to automatically transform the abstract user interface model into a platform-specific presentation and to manage the user's input. In order to achieve the mapping from abstract user interface objects to specific ones, each mobile interaction service has knowledge about the platform specific capabilities (e.g. user interface elements, supported interactions modalities) of the mobile device it is running on. To avoid developing an individual execution environment for each single device, the current prototype environment is based on the J2ME device classification [17] which is applicable to most of today's mobile systems such as notebooks, PDAs and mobile phones (cp. figure 4). Thereby, device-specific adaptations can be minimized and do mostly affect the runtime environment for certain interface modalities, such as audio and video representation. Of course, the interaction service can as well be realized in any other programming language supported by the mobile device.

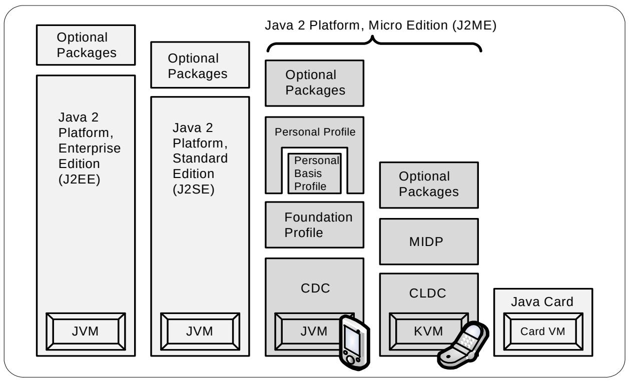

Figure 4: J2ME architecture and resulting device classification [17]

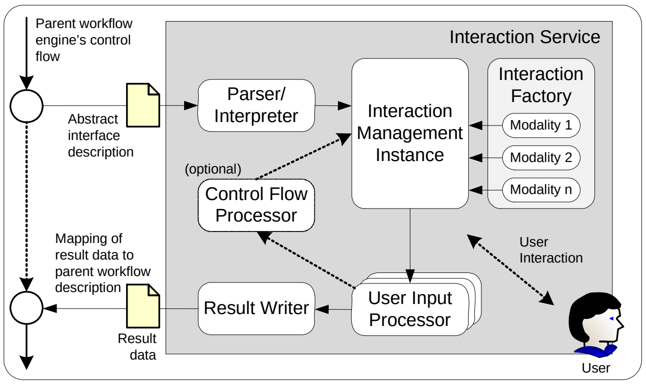

Figure 5: Schematic diagram of the prototypical interaction service

Because the concrete implementations differ slightly w.r.t. device classes and modalities, figure 5 gives a schematic overview of the overall architecture. Once an abstract description is received by the interaction service, the description is parsed and interpreted in order to create an internal model which acts as a basis for the *Interaction Factory* to create a modality-specific interaction dialog. This approach is flexible enough to incorporate any events that happen at runtime and hence allows adapting the interaction to the current context and system specifics by choosing an appropriate modality. The *Interaction Management Instance* is responsible for presenting the created dialog to the user and accepting any user input. Afterwards, the user input can optionally be fed into the interaction service's internal *Control Flow Processor* to realize a sequence of interactions without intervention by the workflow engine. Finally, when the user completed the interaction, his input is encoded by the *Result Writer* and passed back to the workflow engine to be integrated into the overall process description.

4.3. Integration

Due to its service-orientation, the mobile interaction service can be plugged into existing workflow management systems or can be integrated as an invoked application, as e.g. considered by the WfMC [6]. In order to show the feasibility of the presented approach the platform-independent interaction model and its corresponding interaction service have been applied to the workflow system as presented in the use case.

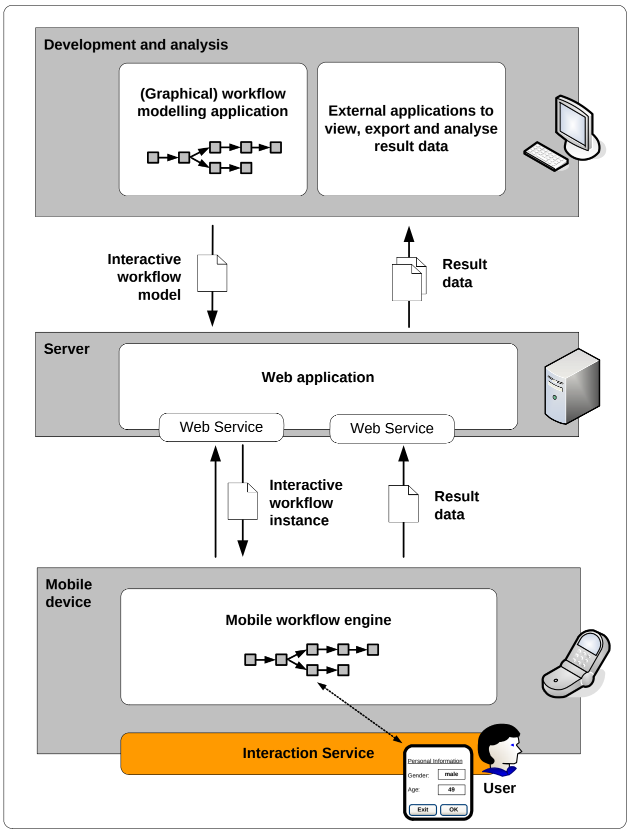
Development and analysis
(Graphical) workflow modelling application
External applications to view, export and analyse result data
Interactive workflow model
Result data
Server
Web application
Web Service
Web Service
Interactive workflow instance
Result data
Mobile device
Mobile workflow engine
Interaction Service
User

Figure 6: Global system overview of the mobile market research workflow distribution

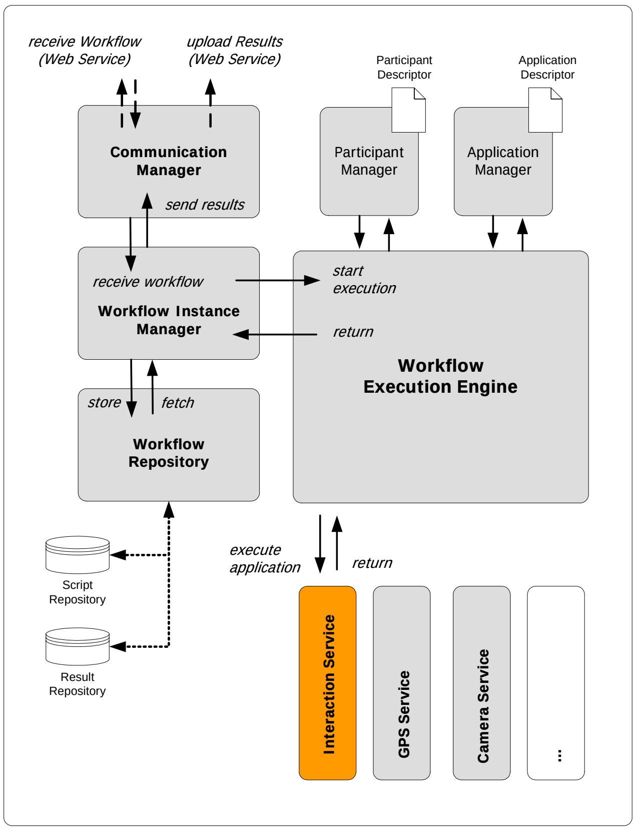

Figure 7: Schematic diagram of the mobile market research workflow engine

As introduced, one of the main goals is to cover and support the entire lifecycle of electronic surveys running on different mobile phones. The corresponding lifecycle starts with modeling the survey, continues with its distribution to mobile devices, goes on with its execution, and finishes with returning the result data to the market research office (cp. figure 6). The first step includes modeling the interactions in the abstract format which is usually performed on a stationary system, such as a personal computer, and supported by modeling tools such as an XML editor or TERESA [9]. As a result, the mobile workflow is described in an XML document containing the control flow logic (here in an XPDL [19] derived language format) and the user interactions specified valid to the abstract interaction model's XML-schema. In the second step, the workflow description is stored on a web server, from where it can be distributed to mobile devices by using a web service. As the web service is accessible from mobile devices, everyone participating in the research can download the workflow description, store it in a local workflow repository and pass it to the mobile workflow execution engine once it shall be executed.

Each time an interactive activity is discovered, the workflow engine calls the interaction module, which produces an appropriate interface representation, collects the user's input data and then returns the control flow back to the workflow engine. Figure 7 shows an overview of the workflow engine's architecture and the integration of the mobile interaction service. As the workflow description specifies both control and data flow, mobile devices can execute the workflow autonomously and no connection to the server is required during runtime. When the mobile workflow has finished, its results may be stored on the mobile device until an appropriate network connection is available. As the last step of the workflow lifecycle, the mobile device returns the results to the web server, again using a web service. Finally, the results of the mobile workflow can be downloaded from the server, can be further processed, or they may trigger other applications.

4.4. Implementation Evaluation

The implementation and integration of the interaction service is straight forward. Two versions have been realized: one for PDAs (using J2ME's CDC) and another one for resource constrained mobile phones (using CLDC accordingly). The prototypes have been developed and tested using the emulators of Sun's Wireless Toolkit as well as Nokia's SDK, which are able to simulate a PDA and a mobile phone respectively, offer support for application profiling and configuration options to change device and infrastructure properties (e.g. processing speed or network bandwidth). As these emulators use different renderings for user interfaces, the mapping of abstract interaction objects (AIO) to concrete interaction objects (CIO) could be optimized. By finalizing certain milestones, the application was also deployed and tested on real devices. The result shows that, due to vendor-specific interpretation of concrete interaction objects, the visual user interface differs slightly from device to device and emulator to emulator. With respect to the layout of components, this is not a problem. A more aggravating factor is, however, the vendor-specific mapping of navigation keys and menu placements which, in some cases, hinder an intuitive navigation (e.g. use left and right buttons for back- and forward navigation).

As some mappings of AIOs to CIOs were expected to be quite resource-intensive, the actual prototype implementation showed the opposite: Although performance issues were not explicitly considered, parsing of the abstract process description (using kXML) as well as the creation of AIOs and the mapping to corresponding CIOs are performed nearly instantly, i.e. within parts of a second. Slight performance drawbacks only appear when large binary data chunks (e.g. audio data or an image taken by a phone's camera) shall be stored as results in the process description. In this case, binary data is Base64-encoded, which takes less than 5 seconds (for a 2 megapixel image) on the mobile phone, but approximately 30 seconds in Sun's emulator on a PC. Apart from this, users are not aware of the on-the-fly creation and composition of user interfaces.

The interaction service itself has been realized as a self-contained component which can either be attached to an existing workflow engine or used as a lightweight stand-alone system. The same service wrapping concept has also been applied for accessing the mobile phones local resources, (e.g. storage, camera, GPS) in such a way as to enable straightforward access to these services from the workflow execution engine, as depicted in figure 7.

5. Conclusion and Future Work

This paper argues that mobile workflow management systems would benefit from a one-for-all approach to design modality-independent user interactions. Therefore, an abstract interface description model is introduced which allows the specification of user-centric workflows without knowledge of the executing device type or its environment. To interpret and to process interface descriptions resulting from this model, a corresponding service-based interaction module has been developed. To show its applicability, a prototypical realization of this module has been integrated into an existing mobile workflow management system, enabling the integration of user interactions with other interactive applications such as the internal camera of a mobile phone. Furthermore, if more than one interaction modality is available (e.g. video or audio), the most appropriate way of interaction can be chosen dynamically and in dependence of the current context situation.
Future work includes solutions for the selection of appropriate participants depending on their context as well as support for automatic workflow distribution. Therefore, advanced management functionalities for mobile information retrieval are required, e.g. regarding the location or current workload of potential workflow participants. It is currently evaluated, whether mobile web service technologies can be applied to provide such informational services for each registered mobile device while respecting the privacy requirements of the mobile user. Such an approach could finally facilitate the definition of the overall business process, e.g. by involving the mobile participant selection procedure in a superordinated process description (e.g. WS-BPEL) to integrate and execute such tasks automatically.

6. References

[1] Agrawal et al., BPEL4People Specification 1.0, https://www.sdn.sap.com/irj/sdn/bpel4people, 2007.
[2] Agrawal et al.: WS-HumanTask Specification 1.0, http://download.boulder.ibm.com/ibmdl/pub/software/dw/specs/ws-bpel4people/WS-HumanTask_v1.pdf, 2007.
[3] Coenraets: An overview of MXML, Adobe Inc., www.adobe.com/devnet/flex/articles/paradigm.html, 2004.
[4] Goodger, Hickson, Hyatt, Waterson: XML User Interface Language (XUL) 1.0, Specification, http://www.mozilla.org/projects/xul/xul.html, 2007.
[5] Hackmann, Haitjema, Gill, Roman: Sliver: A BPEL Workflow Process Execution Engine for Mobile Devices, in: Proceedings of 4th International Conference on Service Oriented Computing (ICSOC), 2006.
[6] Hollingsworth: The Workflow Reference Model, WFMC-TC-1003 1.1, 1995.
[7] Kunze, Zaplata, Lamersdorf: Mobile Processes: Enhancing Cooperation in Distributed Mobile Environments, in: Journal of Computers, 2(1):1-11, 2007
[8] Microsoft Corp.: XAML, msdn.microsoft.com/en-us/library/ms747122.aspx
[9] Mori, Paterno, Santoro: Design and Development of Multidevice User Interfaces through Multiple Logical Descriptions, in. IEEE Transaction on Software Engineering 30, Nr. 8, 2004
[10] Müller-Wilken, Wienberg, Lamersdorf: On Integrating Mobile Devices into a Workflow Management Scenario, in: Proc. 11th International Workshop on Database and Expert Systems Applications (DEXA), 2000.
[11] OASIS: UIML Standard, www.oasis open.org/committees/tc_home.php?wg_abbrev=uiml, 2007.
[12] OASIS: WS-BPEL Spec. 2.0, www.oasis-open.org/committees/tc_home.php?wg_abbrev=wsbpel, 2007
[13] Pajunen, Chande: Developing Workflow Engine for Mobile Devices, in: Proceedings of the 11th Enterprise Distributed Object Computing Conference (EDOC), 2007.
[14] Pajunen, Chande: ActiveForms: A Runtime for Mobile Application Forms, in: Proceedings of the International Conference on the Management of Mobile Business, 2007.
[15] Paterno: Model-based Design and Evaluation of Interactive Applications, Springer 2007
[16] Satyanarayanan: Fundamental Challenges in Mobile Computing. In Proceedings of the 15th ACM Symposium on Principles of Distributed Computing , 1996.
[17] Sun Microsystems, Inc.: J2ME Technologies Overview, Datasheet Overview Java 2 Platform Micro Edition, 2002.
[18] W3C: The Forms Working Group, www.w3.org/MarkUp/Forms, 2008
[19] WfMC: XML Process Definition Language, Version 2.0. Specification WFMCTC-1025, 2005.