# A Workflow Definition Language for Business Integration of Mobile Devices

Martin Werner[1], Stephan A.W. Verclas[2], and Claudia Linnhoff-Popien[1]

[1] Mobile and Distributed Systems Group
Ludwig Maximilian University Munich, Germany
martin.werner,linnhoff@ifi.lmu.de,
http://www.mobile.ifi.lmu.de/
[2] T-Systems International GmbH, Germany
Stephan.Verclas@t-systems.com,
http://www.t-systems.de/

**Abstract.** The integration of mobile devices into business processes is a challenging but promising task. We designed a system that shows the possibility of constructing software systems that accomplish the difficult achievement of a system that simplifies tasks for businesses as well as for their customers. We designed a very simple and straight-forward workflow definition language that has the advantage of allowing source-code generation for arbitrary mobile platforms. With this language we assembled a system that is able to support insurance case documentation with context-aware input wizards and cross-platform software. It is possible to take different actions in different surroundings and to integrate mobile sensor data (GPS, cell-location, audio, compass, ...) into workflow management.

**Keywords:** Workflow management, Workflow management systems.

## 1 Introduction

Workflow modelling and management has become a very important tool for business management and organization. With this paper we show that workflow definition can even be used as a tool for structured communication schemes with customers. We assembled a system with which basic workflow can be specified and automatically translated into source code for a mobile application. In this way it is possible to integrate the process of documenting some event into the surrounding workflow in a natural and automated way. We solved the problems of diversity arising from mobile devices by specifying a language that can be translated into different programming paradigms without difficulties. By using workflow constructs it is even possible to design advanced input forms that make use of context information. An example might be to use audio input only in cases where the surrounding noise is not too high and to give screen input possibilities as an alternative in the other case. The pay-off of such a system is the correctness and completeness of the document as well as the adaptivity of the application.

Y. Cai et al. (Eds.): Mobilware 2010, LNICST 48, pp. 353–362, 2010.

The scenario for which we provide a solution with this paper is based on a company and an individual which have seldom but important and complex communication demands. As a main example we chose an insurance company. Insurance companies usually do not have regular communication with their customers. In an insurance case however the customer (insurance holder) and the business (insurance company) have to follow pretty complicated workflows which at least the customer is not familiar with. The complexity of the workflows arises from the need to check the insuree's information (address, phone number, ...), to exchange complex information (what, when, where, why) and the need for decisions such as whether expert assessment are needed or how to provide an appropriate solution (e.g. immediate reparation vs. taking over costs for car rental and transportation).

For our application let us assume that the insurance case happens in a substantial level of provision of infrastructure for mobile Internet applications to work and that the insurance holder is equipped with a mobile phone providing Internet access. The idea is now to support the classical workflow for an insurance case - which usually starts with an insurance holder calling the insurance company's call center - with adaptive and context-aware software for the mobile device of the insurance holder.

The insurance company's call center agent will ask for the type of mobile phone, the mobile phone number and the level of familiarity with installing and using mobile applications. In cases where the insurance holder is capable of installing specialized software and using applications which he is not familiar with, the call center agent can automatically create a mobile application tailored to the customers device and situation which will support the complete documentation workflow of the insurance case. He will then send a download link for the software via SMS, MMS or email directly onto the insurance holder's device.

In this way the insurance company can make use of the full power of the mobile device and support the insurance case documentation with sensor data, video, audio and photo as well as GPS position information. In this way the insurance gets a very clear and complete record of the insurance case and the insurance holder gets a simple and clear way to explain what has happened and what he would expect from the insurance company.

One of the major aims of such a software supported workflow is the reduction of the number of expert assessments which are made because of incomplete or incorrect insurance case documentation.

Figure 1 gives an example of an insurance case which is documented using our specialized mobile software. The insurance holder has had a car accident. He calls the insurance company for help. The call center agent of the insurance company finds out that the insuree has a suitable mobile phone and the needed background on using mobile applications. They discuss the insurance case such that the call center agent can decide what has to be done for documenting the case. The call center agent constructs an application which the insuree can use to document the insurance case. This software package is then provided to the customer over the air. The insurance holder then uses the application to

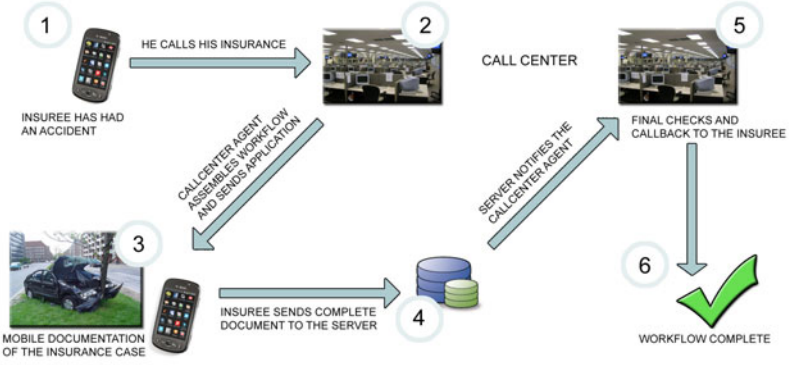


**Fig. 1.** An example case

compile a complete documentation of the insurance case. In cases where he has a question, he uses a help button which directly connects him to the same call center agent. The data he accumulates is sent to the call center agent who can immediately see the photos and videos the insurance holder collected. Once the information has been uploaded, the call center agent gets notified and will call the the insurance holder for further steps.

## 2 Related Work

Recently workflow management has become one of the most valuable information technological tools for business management. Most systems in this area try to support business by modelling the complex relations between business tasks (parallelity, concurrency), business demands (pre- and postconditions of tasks) and business resources (manpower, computing resources, deadlines) into one global language. The complexity of such a system is of course the biggest pitfall in this area. It is not easy to show that a dynamically changing and distributed business process model keeps complete and well-defined for all input and free of dead-locks and even contradiction.

To remedy these restrictions, a business process is seldom modelled in its full complexity. Though there can be areas where a workflow definition is not easy to model and a description of data and data requirements might be better, we want to concentrate on the subpart of workflow specification. The term workflow can be understood from various different perspectives [8]. If we are talking about workflow in the following, we always mean the *control-flow* perspective of workflow. To make this explicit: A workflow description in our sense describes the tasks and their execution ordering and logic where a task is an atomic unit of work. There have been several languages in discussion ranging from simple languages with few features to complex languages with many features but the difficulties arising from faults.

Process definition languages have been there for a long period of time. Examples are the Process Interchange Format [1] which was later incorporated

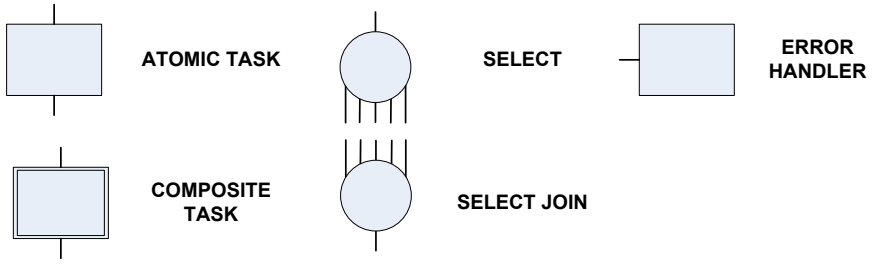


**Fig. 2.** Symbols used for Workflow Definition

to NIST's PSL [2] which also has an XML mapping [3]. Business modelling languages always contain mechanisms to define processes. The Business Process Management Initiative (BPMI) tried to establish a standard language BPML for design and specification of business processes since 2001. The BPMI joined the Object Management Group (OMG) in 2005. The Workflow Management Coalition proposed another XML-based language XPDL [4] which aims at business interoperability. A systematic analysis of this language and of possible workflow patterns is [5]. A newer open initiative is YaWL (2005), a language that tries to solve issues which pop up due to synchronization problems and non-local withdrawals that arise in complex business models organized as Petri nets.

The overall aim of these languages is the adoption of automation and controlling to general business management. One tries to put all business logic and side-effects into a global model which can be understood and optimized with automated systems.

All these languages have in common their high level of complexity which makes them difficult to use in mobile and distributed ([7]) environments. Many of the enhanced features of current process definition languages are not needed for our scenario. Hence we decided to define a suitable subset of such languages which can be used to model the workflow of an intelligent and self-adjusting context-aware application helping the users filling out complex forms.

## 3 A Workflow Definition Language for Mobile Environments

From the discussion of the previous section we now derive requirements and limitations for a language tailored to defining complex workflow without concurrency, mainly supporting user input. It is common to use a set of symbols to define workflows. Symbols are the atomic parts of the language and can be semantically arranged as a graph where each symbol can have a restriction on the number of incoming and outgoing edges. Figure 2 gives an overview of the atomic symbols of our workflow language.

These symbols can be arranged into a tree-structure. The application will then walk through the tree and take appropriate action for a specific task. The conditions that we need range from a set of device properties which are only

known at runtime (e.g. camera resolution, GPS-accuracy) to some process events (failed to get a position).

In favor of maximal simplicity we have collected the following symbols for our language. Atomic tasks are parts of work that can not be split up. Examples are taking pictures, video, input forms, web browser session, phone calls etc.). These atomic tasks can be arranged into lists which have a defined ordering (e.g. wizards). For context-awareness and adoption we decided to only allow those splits where exactly one of the possible choice is taken (XOR-split) and none of the connected atomic tasks has been executed before. For error management each atomic task can set a pointer to another task which shall be executed on error. In this way we are able to model the following two basic error handling mechanisms: A "On Error Resume Next"- strategy, where a task which has reported some error condition is stopped and the workflow continues with the next atomic task or an Error-Handling task which can be set for a given atomic task. These error handling strategies make it possible to have a default strategy of e.g. calling the call center-agent to discuss the problematic situation but also the flexibility to react on a problem with a specific alternative (e.g. if there is no GPS fix we can use a network location service and a map).

Most process definition languages also contain constructs for circuits and concurrent task execution such as AND-splits, XOR-splits, AND-joins and XOR-joins. We decided not to allow any of these complex constructs because of the difficulties with synchronization points (see [6]). The most important problem with these constructs is that it might be unclear at design time as well as at running time whether a synchronization is needed (i.e. waiting for several branches to complete) or whether synchronization is not needed or intended. This will make the application behavior unpredictable and hence has implications on usability. A concrete example would be the case where the application on the mobile phone of the user does not work correctly or the user is not able to use it correctly and then calls back the call-center agent through a help button. The call-center agent should have a complete understanding of the workflow to be able to help in this situation.

There are of course workflow patterns which are not possible or very difficult to model without the constructs we removed. But the language shall only be used for a small part of some bigger workflow where the non-mobile part of the workflow can have all these complex constructs.

The workflow which can be defined with our proposed language is free of circuits except for the case of error handlers pointing back and forth between two atomic tasks that fail. To remove this case, we insist that the depth of error handler execution has a fixed maximum and that a global error handler will terminate such situations.

We decided to describe the workflow in an XML-file format. We are aware of the fact, that the usage of XML will lead to very much overhead, but the reason for using XML is, that for the prototype the benefit of using XML lies in the fact that most mobile platforms support XML-parsing and that XML has a

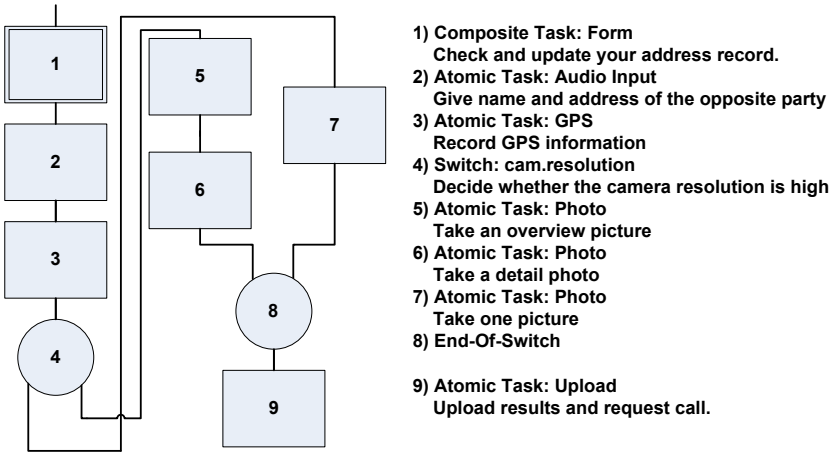


**Fig. 3.** An Example Workflow

tree structure and is extendable. For a product, the complete language should of course be packed into a more efficient data format.

For each symbol we introduce an XML-element. Each element must have an attribute **id**. Each symbol can have an attribute **errorhandler** which contains either the **id** of the error-handler or the special value **next**. If the error-handler is not specified for a task but some parent element has set it, then it is inherited. An atomic task must not have any child-elements. A composite task can have any workflow object as a child element. A switch has several **case** elements which define conditions on the execution of a case. These cases are checked in the order they are specified and the first one whose condition attribute evaluates to true is used. The error-handler of the switch statement is used if none of the case conditions evaluates to true. A switch element has to be closed in the same composite task (this is automatically enforced by XML).

Figure 3 shows an example workflow. The error handlers have been omitted in the graphical representation. The application being defined will first ask the user for some personal information for which presets are given. Then the contact data of the opposite party shall be recorded in audio. Once this basic information is available the workflow tries to locate the phone using GPS. The following switch decides whether the camera of the phone has enough resolution (e.g. 5 mega-pixels) to document the case using one photo or whether a detail picture is needed. Recall, that switches need not be complete and may have several cases which evaluate to true. Only the first one of those cases which have a condition that evaluates to true is being executed. If there is no such task, then the error-handler of the task is invoked.

The XML description of this example workflow is as follows:

```
<?xml version="1.0" encoding="utf-8"?>
<workflow errorhandler="9">
```

```
  <compositetask id="1">
    <task id="10" type="inputtext"
          preset="Martin Werner">Name:</task>
    <task id="11" type="inputtext"
          preset="Some Street 20">Street:</task>
    <task id="12" type="inputtext"
          preset="D-81549 Munich">ZIP / City:</task>
  </compositetask>
  <task id="2" type="inputaudio">Say name, address and
                  phone number of the opposite party!
  </task>
  <task id="3" type="inputgps"/>
  <switch id="4">
    <case condition="phone.camera.resolution >= 5">
      <task id="7" type="inputphoto">Please take a
                           photo of the situation!
      </task>
    </case>
    <case condition="true">
      <task id="5" type="inputphoto">Please take an
                    overview photo of the situation!
      </task>
      <task id="6" type="inputphoto">Please take a
                 detailed photo of the main damage!
      </task>
    </case>
  </switch>
  <task id="9" type="uploadandcall"/>
</workflow>
```

## 4 Prototype

To show the feasibility of our approach we have assembled a prototypical implementation. This implementation consists of three parts: An agent application, a server infrastructure and a mobile application.

For the call-center agent we developed an application with which he can define properties of the mobile phone as well as define a specific workflow using drag-and-drop with predefined tasks (either atomic or composite). We used a car accident example as our show-case.

The server infrastructure implements a HTTP interface to facilitate the communication between agent application, database and applications. The main format for data exchange is XML or HTTP file uploads. The decision for this comes from the fact that nearly all devices with Internet access have possibilities to perform file uploads and have XML support built in.


(a) Start screen (b) Complete incident

**Fig. 4.** Screenshots from the agent application

### 4.1 Agent Application

The agent application is implemented in Flash and can be used to create new database records for an incident, to create the mobile application, to send the mobile application to the customers mobile phone and to read out the results of the workflow. A basic ticketing structure has been introduced to organize these records. The state of such a ticket is

- new, if it has been created
- open, if the mobile application has been downloaded
- complete, if the mobile application has uploaded a result
- closed, if the call center agent has checked the data

This application creates the XML definition of the workflow and uploads it to the server. It is also capable of sending a download link via SMS or email directly to the mobile phone of the insuree. The phone number of the calling person is extracted automatically where possible.

Figure 4(a) shows a typical view of the agent application and 4(b) shows the user interface for a complete incident.

### 4.2 Server Infrastructure

The server component is implemented in PHP and MySQL. It provides an interface to add an attachment file to some given incident as well as an interface to upload the results of basic questions as an XML file. The applications will create the answer and upload an XML description of the results of the workflow which it then extends with attachments (photos, videos, audio, etc.) using HTTP file upload.

### 4.3 The Android Application

For the example case we assembled a generic Android application which can be compiled with an incident id and an session id used for authentication. Supported atomic tasks are

- Input text
- Take a photo
- Take a video
- GPS position
- Coarse network location
- Location selection on a map
- Call a phone number
- Upload and request call-back

A composite task which only contains input text atomic tasks is accumulated to one scrollable form. Due to the structure of an Android application we were able to directly map an atomic task to an Android activity and to implement the workflow logic in the main activity which posts intents for each atomic task. In each activity we show a help button which directly calls the call center agent.

This Android application shows that it is possible to integrate modern mobile platforms in a very intuitive way into business processes.

### 4.4 Integration of Other Mobile Platforms

Other mobile platforms can be integrated very easily. Once the agent has published the workflow description to the server, the server can automatically create source-code from templates for each platform and compile and sign them. This is possible for all mobile platforms which have support for downloading and installing software over the air. It is even possible to implement very basic workflow as a web application and hence provide help and access for all users. Though we do not have access to sensor data, photo, etc. in this case, our application still helps to organize and explain the workflow for the specific user.

## 5 Outlook

With this paper we have presented a new approach and a case study on integration of mobile phones of customers into the workflow of a business. We have solved the problem of the huge differences between the individual mobile platforms by introducing an abstract and simple language to describe the functionality in a way that can easily be mapped to source-code for different platforms. In this way it is possible to use different mobile platforms with the same logic. Most of the automatic source-code generation can be done with XSLT and some script that assembles the sources, compiles the program, signs it and distributes it via a web server.

We showed that the usage of mobile phones can help to follow complex workflows. The benefits are clear. The company can assure that a specific workflow is

completed immediately and satisfactorily and the customer does not suffer from misunderstandings and time-consuming correspondence. Interesting features for such an application could be to have a life-chat with the call center agent while the application is active. This feature is however difficult to implement across all major mobile platforms. We plan to investigate this question further.

An application as described before is not limited to the documentation workflow we aimed at. We think that it could be an interesting question whether it is possible to support business management by mobile workflow management using mobile phones. We have shown that the diversity of platforms is not that a big problem in this area.

## References

1. Lee, J., Grunninger, M., Jin, Y., Malone, T., Tate, A., Yost, G., et al.: The PIF Process Interchange Format and Framework Version 1.2. The Knowledge Engineering Review 13(1) (March 1998)
2. Schlenoff C., Gruninger M.,Tissot F.,Valois J.,Lubell J.,Lee J.: The Process Specification Language (PSL) Overview and Version 1.0 Specification (1999)
3. Lubell, J., Schlenoff C.: Process Representation Using Architectural Forms: Accentuating the Positive. In: Proceedings of the Markup Technologies Conference 1999 (1999)
4. Workflow Management Coalition: Workflow Process Definition Interface XML Process Definition Language (XPDL), WfMC Standards (2001), `http://www.wfmc.org`
5. van der Aalst, W.: Patterns and XPDL: A Critical Evaluation of the XML Process Definition Language (2003)
6. van der Aalst, W., ter Hofstede, A.H.M.: YAWL: yet another workflow language (2004)
7. Dong, G., Hull, R., Kumar, B., Su, J., Zhou, G.: A framework for optimizing distributed workflow executions. In: Connor, R.C.H., Mendelzon, A.O. (eds.) DBPL 1999. LNCS, vol. 1949, pp. 152–167. Springer, Heidelberg (2000)
8. Jablonski, S., Bussler, C.: Workflow Management: Modeling Concepts, Architecture, and Implementation. International Thomson Computer Press (1996)