Creating Mobile ad hoc Workflows with Twitter [*]

Martin Treiber, Daniel Schall, Schahram Dustdar
Distributed Systems Group, TU Vienna
Argentinierstrasse 8, A-1040 Wien
{lastname}@infosys.tuwien.ac.at

Christian Scherling
ikangai solutions
Treustrasse 59/5/20
A-1220 Wien
chs@ikangai.com

ABSTRACT

The wide-spread adoption of powerful mobile devices allows for the design of workflows while on the move. In this paper, we leverage established SOA principles towards mobility in design and execution. For this purpose, we introduce a lightweight Service workflow model that is tailored for the needs of ad-hoc creation and mobility. We show how sensor data from mobile devices can be included during the design of Service workflows and we address crowd sourcing aspects for the deployment and execution of Service workflows. We present a proof of concept prototype application that supports the creation of Service workflows on mobile devices.

Categories and Subject Descriptors

H.4 [**Information Systems Applications**]: Miscellaneous; D.2.11 [**Software Architectures**]: Service-oriented architecture (SOA)

General Terms

Mobility, Workflows, SOA

Keywords

Mobility, Crowd Sourcing, Workflows

1. INTRODUCTION

Service workflows are one of the key concepts of Service Oriented Computing (SOA) [1]. As of today, there is a large number of languages and tools available that supports the creation of workflows for the reuse of Services. More recently, Service mashups build of Restful Services [5, 17] were proposed as alternative [10] to the SOA stack.

With the wide-spread adoption of mobile devices that provide sufficient connectivity and computation capacities, toolsets emerged that bring SOA concepts mobile devices. However, little attention was paid to the design and creation of workflows *on* mobile devices. In this paper, we describe an approach that allows users to design and consume Service workflows directly on their mobile devices. In doing so, we create ad hoc and personal Service workflows which are designed and executed spontaneously in the social context of the workflow designer. In particular, we build upon the Human Provided Service approach [11] that allows us to abstract from concrete workers and to integrate them into Service workflows - building a mixed system of human provided and software Services [14].

The rest of the paper is organized as follows. We introduce the main challenges for the creation of mobile ad hoc Service workflows in Section 2. In the following sections we discuss our approach in and present our prototype application in Section 4. We conclude the paper with related work and an outlook for future work.

2. MOBILE WORKFLOW CHALLENGES

Lightweight Workflow Model. Workflow languages like BPEL provide a large set on functions for the execution of business processes on enterprise infrastructures. In mobile environments, we operate on a different scale in terms of infrastructure. Thus we need to integrate mobile services with a lightweight coordination and communication infrastructure that is tailored to mobile devices.

Ad hoc creation of a workflow in a mobile environment. Workflow editors require at least a laptop to work, a limiting factor for mobility. In mobile environments, creating ad hoc workflows that includes context information, walking around with a laptop is not practical. In addition, laptops might do not provide the necessary sensor array to capture context information (GPS data, camera). Consequently, we need a tool that can be used on mobile phones to create the workflow in situ and be able to integrate sensor data (GPS, pictures) during the design of the workflow.

Working the Crowd. In contrast to traditional workflow systems which are used in static settings with a small number of participants, we have a considerably larger number of participants. For example, we can use local volunteers that are connected over a social network in a mobile Service workflow. This task requires the ability to address a crowd consisting of volunteers and local workers in order to perform a specific task.

*The research leading to these results has received funding from the European Community Seventh Framework Programme FP7/2007-2013 under grant agreement 215483 (S-Cube).

3. LIGHTWEIGHT WORKFLOW MODEL

Our prototype supports Tweetflows [15] which is a simple language for the creation of workflows. In this section, we investigate how our proposed Twetetflows programming model address these issues.

Flat graph model. Our proposed process model is a flat graph model with no nesting (with the exception of iteration loop constructs) and no dedicated exception handling in order to keep the development as simple as possible and applicable on mobile devices. Our model contains two structured activities that can be compared with features of other workflow languages such as BPEL. Tweetflows provide structured iteration loops (so called closed sequences) and the ability to include choices (<pick> activities).

Workflow scripting. We follow an scripting approach in which we do not use typing constructs: variables can be directly used without having to declare (their type) first. The output of activities can be accessed with implicit activity variables and we reduced the level of indirection by omitting message typing and Service binding. We directly include URLs to input data and we do not require the specification of external partner links.

Workflow Language. As with most workflow languages, Tweetflows contains two main constructs: activities and links. Activities are units of work and are connected with links that define dependencies between activities (see Table 1). In addition to links between actions, we also support unstructured activities, i.e., a set activities that can be executed in any order. This is also reflected in the Tweetflow editor, which provides a canvas to position arbitrary activities.

Symbol	Description
SR	Service Request
SF	Service Response
RT	Retweet Service Request
LG	Log Service Activtiy

Table 1: Tweetflow Language Primitives.

Tweetflow activities can have a *transitionCondition* which triggers the execution of activities in Tweetflows. We model these similar to Unix pipes and have similar execution semantics: after an activity is completed, the result serves as input for the next activity and triggers its execution (see Listing 1).

Listing 1: Tweetflow Syntax Structure

```
command = metadata {address}
operation"."object [data]{hashtags}{condition}
metadata = "SR"|"SP"|"SF"|"LG"
address = "@"chars
data = url | url-ecodeddata
hashtags = "#"chars
condition=key"="value
key=chars
value=chars | url
```

4. COMPOSING MOBILE WORKFLOWS

Mobility constraints require to have a lightweight editor that can be used on a mobile device and a graphical user interface to support the quick creation of a Service workflow. Our prototype implementation is build on the Android platform and is able to generate complex Tweetflows with a simple to use interface (see Figure 1).

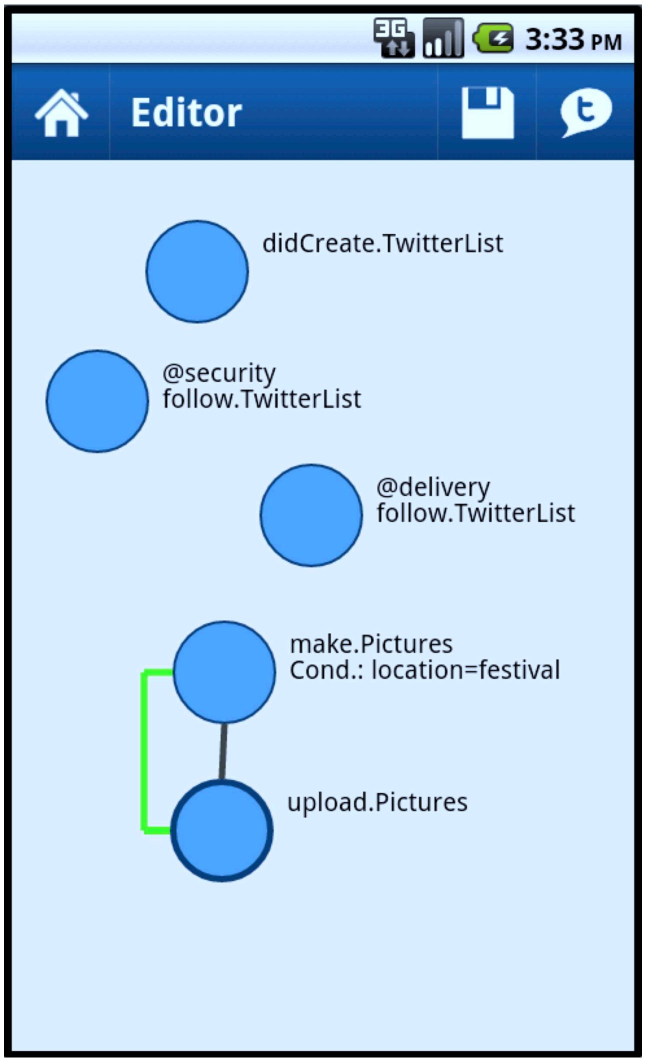

Figure 1: Tweetflow Editor.

We followed the design of traditional workflow editors, which represents activities/actions and their connections as graph. The vertices (dots) represent a Tweetflow command (e.g., a Service Request) and the edges are used to create complex constructs like closed sequences. For the integration of available context information during the design of the Service workflow. we use sensor data that is available on a mobile phone, such as geo location, time, language, gyroscope data or network connectivity (WLAN, 3G, GPRS, EDGE). We decided to add the ability to capture this data directly into the editor: the user has a set of context buttons on the screen that allow for the capturing of context informations.

5. WORKING THE CROWD

By adopting a popular social network platform like Twitter, we tab into the social network of the creator of a mobile, ad hoc Service workflow. Twitter followers receive the Tweetflow specification and can be directly addressed in the

Tweetflow using Twitter's build-in addressing operators. By exploiting the follower structure we are able to distribute the responsibility for the execution of a Tweetflow among the participants of a Tweetflow, i.e., we crowdsource the execution of the Tweetflow.

Tweetflows offer two types of adaption mechanisms that are salient features of collaboratively executing a Service workflow. Firstly, Service instances can be actively changed during the execution of the workflow. Each Service provider can modify the addressee of the tweet by delegating a Service request to another Service provider, other than originally specified. Secondly, during the execution of a Tweetflow, tweets can be added that perform additional actions by the Tweetflow followers.

The aspect of social trust is out of scope of this paper, we refer the interested reader to the work of Skopik et. al [14] for a detailed discussion on this matter.

6. RELATED WORK

TurKit [4] is a crowd computing framework based on MTurk and is closely related to our work. [12] discusses a hybrid human-computer document translation system, but does not focus on the realization as a service-based system. Juszczyk et. al. [3] introduce a middleware for service-oriented communication which runs on mobile devices. Architectures for Mobile Web Services [7] aim at providing alternative representations other than XML-based SOAP and fast communication transport options for mobile Web Services [8, 2]. The approach presented in [9] uses aspect oriented programming to facilitate the access to Services from mobile devices. [16] describes an infrastructure which is based on the Jini Surrogate Architecture Specification. A mobile SOA-based architecture based on J2ME is discussed in [6] which minimizes the traffic between mobile devices. Singh et. al[13] investigate the use of short messages for communication purposes between mobile devices in asynchronous Service invocation.

7. FUTURE WORK

In future work, we will study the social interface to the crowd in greater detail and introduce a social programming layer for mobile apps that integrates Tweetflows directly into mobile applications. For this purpose we will extend the Tweetflow language with additional constructs that facilitate this kind of integration. We are going to extend our prototype with additional features and conduct user studies on the usability.

8. ACKNOWLEDGEMENTS

We thank our students Matthias Neumayr and Martin Perebner for their work on the prototype implementation.

9. REFERENCES

[1] G. Alonso, F. Casati, H. Kuno, and V. Machiraju. *Web Services - Concepts, Architectures and Applications*. Springer, October 2003.

[2] F. Jammes, A. Mensch, and H. Smit. Service-oriented device communications using the devices profile for web services. In *MPAC '05: Proceedings of the 3rd international workshop on Middleware for pervasive and ad-hoc computing*, pages 1–8, New York, NY, USA, 2005. ACM.

[3] L. Juszczyk and S. Dustdar. A middleware for service-oriented communication in mobile disaster response environments. In *MPAC '08*, pages 37–42, New York, NY, USA, 2008. ACM.

[4] G. Little, L. B. Chilton, M. Goldman, and R. C. Miller. Turkit: tools for iterative tasks on mechanical turk. In *HCOMP '09*, pages 29–30. ACM, 2009.

[5] E. M. Maximilien, A. Ranabahu, and S. Tai. Swashup: situational web applications mashups. In *OOPSLA '07: Companion to the 22nd ACM SIGPLAN conference on Object-oriented programming systems and applications companion*, pages 797–798, New York, NY, USA, 2007. ACM.

[6] Y. Natchetoi, V. Kaufman, and A. Shapiro. Service-oriented architecture for mobile applications. In *SAM '08: Proceedings of the 1st international workshop on Software architectures and mobility*, pages 27–32, New York, NY, USA, 2008. ACM.

[7] S. Oh and G. C. Fox. Hhfr: A new architecture for mobile web services: Principles and implementations. Technical report, 2005.

[8] S. Oh and G. C. Fox. Optimizing web service messaging performance in mobile computing. *Future Gener. Comput. Syst.*, 23(4):623–632, 2007.

[9] G. Ortiz and A. G. D. Prado. Improving device-aware web services and their mobile clients through an aspect-oriented, model-driven approach. *Inf. Softw. Technol.*, 52(10):1080–1093, 2010.

[10] C. Pautasso, O. Zimmermann, and F. Leymann. Restful web services vs. "big"' web services: making the right architectural decision. In *Proceeding of the 17th international conference on World Wide Web*, WWW '08, pages 805–814, New York, NY, USA, 2008. ACM.

[11] D. Schall, H.-L. Truong, and S. Dustdar. Unifying human and software services in web-scale collaborations. *IEEE Internet Computing*, 12(3):62–68, 2008.

[12] D. Shahaf and E. Horvitz. Generalized task markets for human and machine computation. In *AAAI*, 2010.

[13] R. Singh, S. Mishra, and D. S. Kushwaha. An efficient asynchronous mobile web service framework. *SIGSOFT Softw. Eng. Notes*, 34(6):1–7, 2009.

[14] F. Skopik, D. Schall, and S. Dustdar. The cycle of trust in mixed service-oriented systems. In *SEAA*, 2009.

[15] M. Treiber, D. Schall, S. Dustdar, and C. Scherling. Tweetflows: flexible workflows with twitter. In *Proceeding of the 3rd international workshop on Principles of engineering service-oriented systems*, PESOS '11, pages 1–7, New York, NY, USA, 2011. ACM.

[16] A. van Halteren and P. Pawar. Mobile service platform: A middleware for nomadic mobile service provisioning. In *Wireless and Mobile Computing, Networking and Communications, 2006. (WiMob'2006). IEEE International Conference on*, pages 292 –299, 19-21 2006.

[17] M. Vasko and S. Dustdar. Introducing Collaborative Service Mashup Design. In *Lightweight Integration on the Web (ComposableWeb'09)*, pages 51–62. CEUR - Workshop Proceedings, June 2009.