End-User Service Composition in Mobile Pervasive Environments

Jacqueline Floch[1], Peter Herrmann[2], Mohammad Ullah Khan[2], Richard Sanders[1], Erlend Stav[1], and Rune Sætre[3]

[1] SINTEF ICT, NO-7465 Trondheim, Norway
{Jacqueline.Floch, Richard.Sanders, Erlend.Stav}@sintef.no
[2] Dept. of Telematics, NTNU, NO-7491 Trondheim, Norway
{herrmann, mukhan}@item.ntnu.no
[3] Dept. of Computer and Information Science, NTNU, NO-7491 Trondheim, Norway
satre@idi.ntnu.no

Abstract. Intelligent objects and devices are becoming part of the environment where people live. The more mobile and pervasive computing becomes, the greater opportunity users potentially have to customize the computing activities that take place around them. For some people the availability of devices and services offers possibilities for tailoring things to exactly what they want. For others, however, this represents a problem: how to manage the complexity? It is neither practical nor economical to use professional software developers for individual tailoring. Thus, we have to provide users with easily operable tools for service composition. The goal of this paper is to highlight the main challenges for a meaningful end-user tool support.

1 Introduction

Service Oriented Architecture (SOA) provides a set of design principles allowing the construction of software systems through the composition of loosely coupled functional entities. Evolving from component-based architectures that improve the reuse of software, SOA goes a step further addressing the problem of business service reuse, meaning that capabilities in a system (or services) may be under the control of different owners. This flexibility both at the engineering and business levels has contributed to applications of SOA beyond enterprise systems, and in particular in the area of mobile pervasive computing that is the focus of this paper. The vision of mobile pervasive computing is that of environments where objects are becoming increasingly intelligent and provide information and services to the user when and where needed. Tailoring things to exactly what the user wants is indeed challenging and requires a good understanding of the user's needs. Why not let the users themselves compose the services according to their needs? Bringing service composition to ordinary people has the clear advantage of putting users in control of their personal and social life in environments where computing becomes pervasive.

While mobile services are accessed from mobile devices, we envision that support for their composition should be provided both on desktop computers and mobile devices. As the screen size of mobile devices remains a major limitation for the design of advanced services, composition support seems more realistic on desktop or tablet computers. However, the service environment of the mobile user changes leading to both new needs and new opportunities. Thus it is relevant to consider that composed services need be adjusted on the fly, using a pocket-size mobile device.

Although some initial research work has been done to show that the concept of end-user composition of mobile services is not just a dream [1], challenges are far from solved. This paper presents some of these challenges, but first we focus on some potentials.

2 Potentials

Putting end-users in control has a great potential, thanks to recent technological advances. We illustrate this through a few scenarios, and discuss enabling factors.

2.1 Scenarios

A service composition approach is applicable to multiple application domains in which end-users are interested in tailoring personalized services and applications without being able to invest in professional expertise. We illustrate this point with two example scenarios from the Ambient Assisted Living (AAL) domain and the eLearning domain respectively. Service composition is also relevant in the tourism domain as illustrated in [2], as well as many others.

Pétanque (boccia): Alice's 75 year old father, Charlie, is starting to forget things, such as the occasional Pétanque appointments with his friends. Alice wants to assist him, while Charlie wishes to manage on his own as much as possible. Using a service composition tool for non-IT professionals, Alice tailors an assistive service to help him. The service connects to the Pétanque group's calendar on the social networking site they use to find meetings. It displays a message on Charlie's TV screen when it is time to go, but only when some of his friends have confirmed that they will attend, because he does not want to turn up alone. Further, the service plans his itinerary using the bus schedule and Google maps, and keeps track of his progress using a positioning service. The composed service also alerts Alice if any problem occurs, for example if Charlie does not follow the planned itinerary.

Discovery game: Alice is a teacher. Taking advantage of her earlier experience with composing assistive services for her father, she gets the idea to set up a discovery game for the local history course. The goal is to let pupils discover various historical places in the city and answer quizzes related to these places.

Using the service composition tool, Alice composes a discovery game service. The service allows her to set up groups of pupils based on the social networking group for the class. Further, the service provides support for defining places of interest using Google maps, associating quizzes to places, collecting answers from groups, following the location of groups using a positioning service, letting groups chat together, and reminding groups when it is time to go back to school. During the supervision of the class in the city, Alice finds out that a local museum provides a new interactive service for explaining historic monuments. Using her mobile phone, she adds the discovered service to her set of relevant building blocks and replaces the dictionary-based service by this new service in the discovery game.

2.2 Enabling technology and collaboration models

In order to facilitate uptake of new ubiquitous service composition technology, it must be made easily available to ordinary users. Currently, iPhone, Android and Windows Phone are the major competitors in the smartphone market. So far, iPhone has been leading innovation and adoption of new technologies, e.g. multi-touch screens, but Android is rapidly catching up, and Microsoft has teamed up with Nokia to release its new smart phones.

One of the things that made iPhone popular is the App Store [3] with more than 350,000 applications available currently. The Android Market is a similar success story with more than 150,000 apps already, tripled just in one year [4]. Marketplaces like these allow users with some programming experience to make programs for everybody else. However, it is hard for the average user to know which of many similar programs to choose. One of the ways to deal with this uncertainty is putting a community feedback and reputation system into place. This is done both as a part of the App Store and the Android Market, and also by many other independent trusted sources running their own discussion boards. However the apps available in these marketplaces are currently stand-alone, and do not lend themselves to easy composition. Current initiatives may change this, e.g. App Inventor [5] and TouchStudio [6].

3 Challenges

3.1 Dynamicity of mobile pervasive environments

Explaining the users what services are about and letting them search for services that fit their needs are common concerns for any end-user service composition approach. Beyond this, composition in mobile pervasive environments has to deal with the dynamicity of environments. Mobile computing environments are highly dynamic: availability and accessibility of devices embedded in the environment change and services available for use appear and disappear and vary in quality. The composition approach should thus enable the users to express the goals they want to achieve rather than what specific services they wish to use. For instance, rather than expressing that a specific bus timetable is needed, the user

may express that his goal is to travel to a specific place at a specific time. This makes the underlying system responsible to search among alternative services. There may exist several services that support similar goals. Service quality is commonly used to differentiate between similar services. In order to select a service dynamically, the user should be able to express the expected service quality. For instance, in the case of transportation, cost and travel duration are typical service qualities. Using a reputation management system may help to support the selection of the most trustworthy services and thus to increase the trust in the resulting service composition. Finally, if no service is available, the user should somehow be informed.

In addition to the dynamicity of the service landscape, the context of using the services may also change influencing user's tasks and needs. The composition approach should therefore provide abstractions for a variety of context information that may influence what compositions are relevant and how they should execute. A challenge of designing services for open environments is that many factors may influence the needs of the users, and it is often difficult to foresee all at design time. Users should thus be given good debugging tools (see next subsection) and the possibility to change the composition of services on the fly.

3.2 Understanding what has been composed

Good tools for individual tailoring and service composition must allow users to easily grasp what they have composed. In the domain of spreadsheet programming, where end-user programming has long since made a breakthrough [7], the very nature of the domain implies that users get immediate feedback on their programming, since the spreadsheet typically recalculates itself after each programming step. This kind of "instantaneous feedback" is difficult in the domain of service composition, where service design and service execution can be days, weeks or months apart in time. Simulation and debugging are essential techniques to provide feedback on what has been programmed. Challenges and the state of the art here are as follows:

Simulation: Users should be able to simulate specific scenarios (e.g. "What would happen if Charlie is not in his living room when a reminder is displayed on his TV?"). In this way the composed service can be simulated and debugged before it causes major annoyances to the users involved. Currently, it seems that existing tools do not support simulation of composed services. We note the following:

- The Ubiwise project provides a first-person-shooter-like 3D interface to simulate ubiquitous computing applications and devices [8]. Developers can use Ubiwise to test new devices' usability before building a first prototype.
- Myers et al. [9] introduced the "why/why not" lines in end-user testing tools and observed that the approach dramatically decreases debugging time.

Debugging: Tools should provide the service composer with an understandable explanation about bugs. With the exception of the domain of spreadsheet programming, there are few results dealing with end-user debugging. Debugging systems usually feature introspection, i.e. one can inspect the state of the software and in some cases the past states. Instead of reporting a state, we envision a reasoning engine that explains to the user which input-events in combination with which rules have caused a certain output event ("Why did it happen?" / "Why did it not happen?").

3.3 Runtime platform concerns

We target a composition solution generic enough to be used by services managed by different runtime platforms. Moreover, for end users to compose their services, intuitive visual notation and tools are needed, while the execution of services depends on possibly varying service implementations and runtime platforms. Some of these issues can be solved by separating the composition of services from their execution. For example, the execution platform can be independent of the visual notation and the composition tool, if the platform only considers the composition expressed as an XML document with a predefined structure described by a meta-model or an XML schema. The support for end-user service composition on mobile devices imposes a number of challenges. We briefly introduce a few of them in the following.

Compatible platform The runtime platform supporting the execution of services should be able to correctly interpret the composition.

Device independent composition: The composition support should cover both desktop PCs and mobile devices. Users of mobile devices would usually like to compose or adjust the composition on their devices, since new needs may arise while they are on the move. On the other hand, it is more convenient to compose on desktop PCs because of richer user interfaces, larger screens and easier browsing opportunities when looking for reusable services. Therefore, compositions created on desktop PCs may need to be edited on mobile devices and vice versa. Fortunately, today's smart-phones are supplied with almost as advanced capabilities as a PC, but the tools and the notations still need to consider device-dependent issues such as screen size.

Updating compositions at runtime: As explained above, end-users will most likely want to adjust their composition on the fly. Such adjustments must be feasible on mobile devices and these should trigger an update of the service usage.

3.4 Diversity of end-users

The "mass-consumer market" nature of mobile pervasive environments holds a great potential for widespread use of end-user service composition, but the great

implied diversity of end-users also adds challenges for design and evaluation. Potential service composers will differ both in their needs, their motivation and in their technical knowledge. This poses challenges in finding the right balance between simplicity and functional richness of the composition environment, and emphasizes the need for supporting the users in learning and evolving their designs at their own speed.

While evaluation in a lab setting can provide feedback on user interface issues of the composition approach, it may only to a limited extent answer how useful it is for the users to be able to compose the services, and what challenges the users have in applying the composition approach to their own needs. More in-depth answers can probably only be found by studying the users in their real usage environment, and over a period of time that is extensive enough to allow the users to learn the technology and to discover when they usefully can apply it.

Acknowledgement. Our research is funded by the Norwegian Research Council through the project UbiCompForAll.

References

1. Yu, J., Sheng, Q.Z., Falcarin, P.: A Visual Semantic Service Browser Supporting User-Centric Service Composition. In: proc. of the 24th IEEE International Conference on Advanced Information Networking and Applications (AINA). IEEE (2010)
2. Floch, J.: A Framework for User-Tailored City Exploration. To be published in proceedings of the Third International Symposium on End-User Development. (2011)
3. App Store. http://www.apple.com/iphone/features/app-store.html
4. Schmidt, E.: CEO Google. In: Mobile World Congress in Barcelona, Spain. (15 February 2011)
5. App Inventor. http://appinventor.googlelabs.com/about/
6. TouchStudio. http://research.microsoft.com/en-us/projects/touchstudio/
7. Abraham, R., Burnett, M., Erwig, M.: Spreadsheet programming. Encyclopaedia of Computer Science and Engineering (2009)
8. Barton, J., Vijayaraghavan, V.: "Ubiwise: A simulator for ubiquitous computing systems design". Tech. Rep. HPL-2003-93, HP Laboratories. (2003)
9. Myers, B.A., et al.: Natural programming languages and environments. Communications of the ACM 47(9). (2004)