Activity Based Client for iOS
Bachelor Thesis

Hildur Uffe Flemberg (270689-3043, hufl@itu.dk),
Niklas Schalck Johansson (121288-2727, nsjo@itu.dk)

Supervised by
Jakob E. Bardram (bardram@itu.dk)

IT University of Copenhagen

Wednesday May 23th

Contents

List of Figures

1   Abstract

TODO

2 Introduction

2.1 Context

In modern times, work practices are often complex entities with respect to the number of resources and work flows involved. The seemingly simple task of doing a standard health check of a patient could easily involve e.g. a doctor, a nurse, a secretary and several documents including the patient record etc. Furthermore, the doctor's office might not be a sufficient location to complete the task - the patient might need to accompany the nurse to the lab to have a blood sample taken, go back to the office to have the doctor test his eyes and initially the doctor might need to get the patients record from the secretary in the reception. Today, computers are heavily used as a tool for optimization when carrying out tasks like these. They solve single tasks on the lowest level like reading e-mails, looking up records in databases and filling out forms very efficiently thus leaving it up to the user to find out for each instance of a work flow which applications, files, documents etc. he needs to prepare to do his job. While this doesn't render the tasks impossible to carry out, it implies some practical overhead in terms of time and mind capacity that moves attention of expert users from things that add value to their work to things that are less important.

2.2 Background

Activity Based Computing is a new paradigm that changes the focus of traditional computing environments from low level tasks like e-mail checking or web browsing to a higher level abstraction in the shape of an activity like "Performing a health check". An activity encapsulates all the smaller tasks and resources that are needed to complete it and provides a manageable unit that can be suspended and resumed, run in parallel with other activities and moved around while it adapts to the new surroundings. The ABC project started in 2003 with it's outset in pervasive computing de- signed for mobile, collaborative and time critical work for clinicians in hospitals. Research on ABC is currently developed and maintained by the ABC research team led by Jakob E. Bardram [1] and is financially funded by The Danish Council for Strategic Research [1]. The research team has developed five versions of the ABC framework, and is currently working on a sixth version. The ABC framework implements services for handling activities and resources. The fifth version in use is a Java based peer-to-peer based on the Aexo infrastructure and is targeted for non-PC devices [1]. The ABC framework will be described in more detail in chapter 2.

2.3 Motivation

With the introduction of touch screen tablet PCs on the market back in 2001 [3], a new family of devices with yet another screen size and touch screen performance matured. Even though the concept of a tablet PC is not a new one, tablets have since 2001 undergone major changes in the sense that they have become common and they have gotten more interactive user interfaces due to improved touch functionality. Bardram [2011] argues that: "Once you move away from the desktop and into a non-office-like environment such as a hospital, the challenges arising from the management of parallel activities and interruption are amplified because multi-tasking is now combined with a high degree of mobility [...]" Given the challenges of mobile work environments, the recent improvement of tablet computers and the fact that the experimentation with iOS as a technically well-suited operating system within ABC has not previously been explored, we will implement an iOS ABC client for the Apple iPad.

2.4 Goals

The purpose of the project is to develop an ABC client for the ABC framework running on the iOS platform with the following goals:

Activity Centered
It must to support the notion of activities.

Activity Awareness
It needs to able to adapt and adjust itself according to its location, mean- ing that the types of resources available and the UI representation is always dependent of the current working context.

Activity Suspend and Resume
It needs to able to save the state of one activity in order to restore and resume another previously suspended activity.

Uniform UI
It must have a uniform UI, meaning that whatever concrete kinds of dis- plays we choose to build these must be the same for any activity as long as the activities are resumed in the same location and under the same conditions.

2.5 Methods

The goals will be achieved through these methods

Investigation - iOS
We will investigate the iOS platform and discuss how we can apply the above mentioned goals. This will be done by following classes on iOS development and reading related articles on the subject.

Investigation - Location tracking
We will investigate what hardware resources is available on the iPad, and discuss which is better for location tracking. We will then discuss how this can be connected to the result of the iOS investigation, and how it will support the ABC principle of activity-awareness.

Implementation
Based on the analysis and discussion of the important elements we will define a list of requirements and implement a client for iOS that supports these requirements. The client will make use of the ABC framework.

Evaluation
We will evaluate the implementation by defining user scenarios that emphasize the mobility chal- lenge in a hospital environment, where a user needs to bring digital resources with them, and test persons complete the scenarios. Afterwards we will have the test persons fill out a questionnaire where they will rate and evaluate the implemented features. Finally we will analyze the results and suggest improvements to our solution.

2.6 Results

TODO

2.7 Overview

TODO

3 Activity Based Computing

In this chapter we will explain the activity-based paradigm further explained. We will explain why adaptation and awareness is important, and introduce two frameworks that will help us create an activity based client.

3.1 Background

As explained in the introduction, activity based computing is a new computer paradigm that moves focus from application based computer interaction into a higher level computational support for human activities. The paradigm has its outset in clinical work on hospitals, and seeks to aggregate resources to activities, instead of specific applications. An example of such an activity could be the development of our proof of concept. Opening an activity will cause the relevant services and resources to become available to the user, and allow to user to more easily switch between activities and all their associated services and resources. Figure 3.1 shows an example of the activity "Proof-of-concept development", its associated services and their resources.
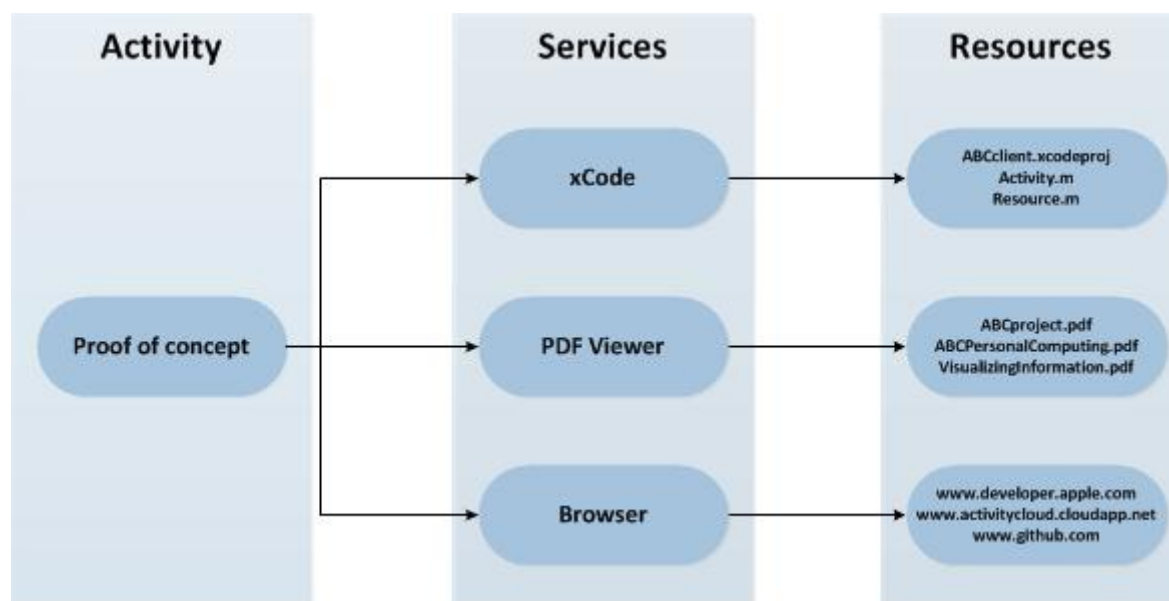

Figure 3.1: *The "Proof of Concept" activity. Illustrates how an activity encapsulates its services and resources*

Bardram [2011] identifies six principles that forms the basis of the activity based computing paradigm, being; *Activity Centered*, *Activity Suspend and Resume*, *Activity Roaming*, *Acitivty Adaptation*, *Activity Sharing* and *Activity Awareness*. Each of these will be further described in the following.

1. **Activity Centered**
   An Activity is a computational unit that encapsulates a set of services and their relevant resources.

An activity therefore encapsulate digital software and data necessary for a user to carry out their work (activity).

2. Activity Suspend and Resume
This allows a user to alternate between several activities and support interruptions that requires the user to perform another task. This is done by suspending the current activity and resuming another.

3. Activity Roaming
This principle enables activities to be stored on an infrastructure, like a server, and allows for activities to be suspended on one device, and then resumed on another, to better support user mobility.

4. Activity Adaptation
An activity can be displayed and handled on very different devices, and should adapt to the resources available at the resumed devices. In this case resources could be CPU power, screen size and network bandwidth.

5. Activity Sharing
Focuses and deals with the collaborative aspect of activities. This principle states that activities are shared among collaborators that appear on a list of participants for any particular activity. If two or more participants engage resumes the same activity they will both be notified and engage in video and audio chat if possible.

6. Activity Awareness
Allows for an activity to be context aware, such that it adapts itself to its current environment and work context. This could be to i.e. adapting the user interface or changing activities and services based on where the device is located.

Implementing all of these features enables computational devices to better support human activities, and allow users to move away from the traditional document -and application centered model on a desktop computer. In this thesis we will focus on how to display activities on the iPad and how to adapt the interface to the orientation of the iPad, how to store activities in an infrastructure and how the iPad can be aware of its surroundings. In this chapter we will further explain the principle of adaptation and awareness.

3.2 Activity Adaptation and Awareness

In many work places it is normal for users to carry out their work in several different locations. As an example it would be natural to consider a plausible scenario for a professor during his day at a university:

> *A professor got a day full of meetings at the University. In the beginning of the day he will have a meeting with the head of a study programme at which he teaches a course. They will discuss several course related material, and course goals, and will require the use of the course website, a document with the goals of the study programme, the exercises used in the course. Later he will have a lunch meeting with a fellow colleague in the cantina, to discuss an idea for a project. During preparation he have found several online resources on the matter, and have made a few designs and diagrams he wants to share. After lunch he got a meeting in his office with a couple of students regarding a bachelor thesis, and needs to review some code written, which includes looking over several source code files, as well as a generated documentation file. Afterwards he got a meeting with a PhD student in his office, regarding his thesis that*

needs review, and also to discuss a certain article found online. During the meeting the search through an article database located online, and find a couple of interesting articles that they save for later use.

This is a thought scenario but it clearly illustrates two things: first of all there is a need to aggregate resources to certain activities during the day for easy access, and second he only need certain activities at certain locations.

The first issue can be handled by using the activity based paradigm as explained earlier, in order to encapsulate resources with different activities. Now the second issue can be handled by filtering the available activities based on where the user is. Now in this case, only four activities have been mentioned but there might be many more than those. There might be activities planned for the rest of the week, and there might even be activities that are not work related. This could potentially sum up to quite a lot of activities and most of them are only relevant when you are in a specific location.

This is where the principle of activity awareness becomes important. Many wireless technologies exists today, which enables devices to communicate with their surroundings, and using these technologies it would be possible for a device to find out about its current whereabouts.
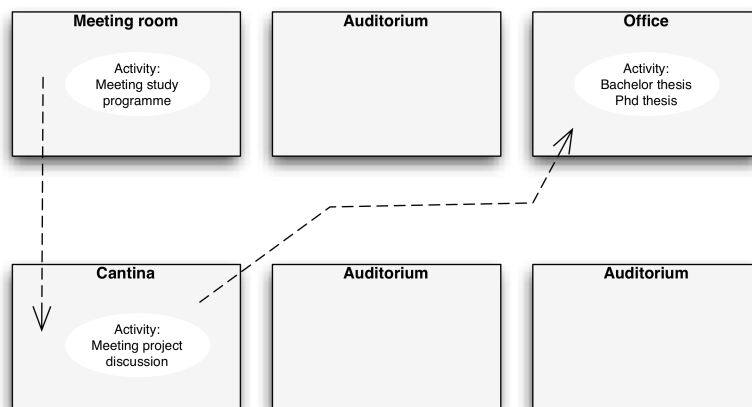

Figure 3.2: *Illustration on how the professor moves between locations in order to carry out his activities*

With current wireless technologies one could use some kind of antennas in different locations and relate those to certain activities. In this way the device could communicate with these antennas and ask where it is right now and based on the activity relations, only show the activities that takes place in this location. We can specify this kind of filtering as *location filtering*. Location filtering is thus a concrete way of handling activity awareness.

With regards to activity adaptation it is interesting to consider the result of the evaluation from Bardram [2009]:

This means that if an activity is roamed from a display with a large resolution (e.g., 1900×1600) to one with a low resolution (e.g., 800×600), a significant portion of the activity's application windows will potentially be left outside the visible area of the display. A related issue arose when clinicians asked for activity roaming between very large devices (e.g., the wall display in the team room) to very small devices (e.g., a PDA or a SmartPhone). In principle, the ABC framework can run on a PDA or a Java-enabled SmartPhone using the J2ME edition. The real challenge, however, is to investigate further what it actually means to roam an activity between two such very different types of devices—especially if we take into consideration that the clinicians saw the small devices as tools for more specific actions within

the overall activity. One possible approach may be to support roaming a subpart of an activity to a small device, instead of roaming the whole activity.

What is really interesting here is that it appears that the same kind of UI is implemented on very different devices, and that a possible solution could be to only handle a subpart of an activity. It should always be discouraged to handle very different devices similarly. It makes sense in standard desktop environments where most computers offer the same screen size and resolution, but when one moves from this environment, as is the case of activity based computing, one should also treat each family of devices differently. This means that PDA's and smartphones should have a distinct UI, tablets should have a distinct UI and so on. One could argue that this would mean a lot of overhead implement different UI's for different devices, but there exist design paradigms that takes this into account, and only require the UI part to changed, and not the rest of the implementation. It is also only natural that the UI is different as these devices would be used very differently as observed in Bardram [2009]:

[. . .]—especially if we take into consideration that the clinicians saw the small devices as tools for more specific actions within the overall activity

So in order to fully make use of activity adaptation, it is important to recognize that each family of devices is different, and should be treated as such, and that their usage is also different. One would probably not replace a wall display with an iPad and hope to achieve the same thing. This is important to keep in mind when designing the UI of the client on the iPad, in order to support activity adaptation properly.

3.3   ABC Framework

In the following, the infrastructure that will be used to store activities will be described in more detail.

The part of the infrastructure that will be used particularly in this project, is the RESTful API. REST, or representational state transfer, is a web architecture that handles transfer of resource states. A resource can be any meaningful encapsulation of data that allows for a client to make request to a server, when data needs to be changed to a new state. Just like when a user is browsing a website, and everytime a new section of the site is visited the state changes by issuing a request and update the local data according to the response. The data itself will be described using a internet media type. In this case the media type is JSON.

Figure 3.3 shows a graphical representation of the RESTful service. The way the service works is by receiving request from a client through the RESTful service. These requests is then received by *Activity Access* which maps the request to an internal function. That function is called on the *Activity Manager*, which looks at the HTTP header of the initial response, to determine what to do with the JSON data. The JSON data is then mapped to an *Activity* object. Based on this Activity object, an *Activity Wrapper* is produced which is placed in the *Data Table(DT)*, and the activity object itself is serialized and placed as a *Binary Large Object (Blop)*, on the service.

So for example when a GET request is issued by a client, the Activity Manger will lookup the Activity Wrapper in the Data Table, to find out where the Activity object is placed and then deserialize it and map it to JSON data and finally send it back as a response.
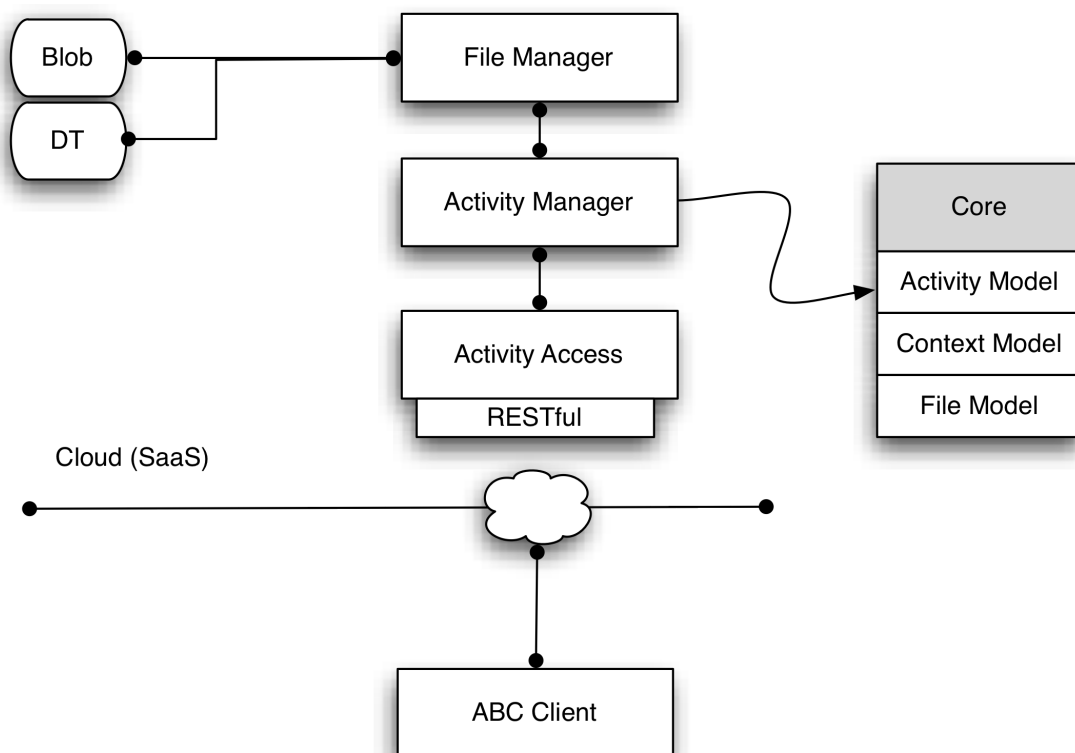
Figure 3.3: *Illustration of the ABC infrastructure*

4 Mobile Activities

4.1 Going from desktop to tablet computers

4.2 Local and cloud computing

4.3 Location tracking

5   Implementation

6 Evaluation

7   Conclusion

Bibliography

A.R.Team. Activity based computing. `http://activitybasedcomputing.com`, 2012.

Jakob E. Bardram. Activity-based computing for medical work in hospitals. *ACM Transactions on Computer-Human Interaction*, Vol. 16(No. 2), 2009.

Jakob E. Bardram. The activity-based computing project. *AAAI Workshop (WS-11-04)*, Techniques and Languages, 2011.

Bardram et al. Support for activity-based computing in a personal computing operating system. *CHI 2006*, 2006.

Bardram et al. Reticularspaces: Activity-based computing support for physically distributed and collaborative smart spaces. *CHI 2012*, 2012.

Luca Chittaro. Visualizing information on mobile devices.

Consolvo et al. Activity sensing in the wild: a field trial of ubifit garden. *CHI 2008*, April 2008.

Froehlich et al.. Ubigreen: investigating a mobile tool for tracking and supporting green transportation habits. *CHI 2009*, April 2009.

Michael B. Skov and Rune Th. Hoeegh. Supporting information access in a hospital ward by a context-aware mobile electronic patient. *Pers. Ubiquit. Comput. 10: 205-214*, October 2006.

Various. Tablet history. `http://en.wikipedia.org/wiki/Microsoft_Tablet_PC`, 2012.