Open Issues in Activity-Based and Task-Level Computing

Jakob E. Bardram and Henrik Bærbak Christensen

Centre for Pervasive Computing
Department of Computer Science, University of Aarhus
Aabogade 34, 8200 Århus N, Denmark
{bardram,hbc}@daimi.au.dk

Abstract. The prevailing computer paradigms do a poor job at meeting their human users at their level of abstractions. Humans organize work and leisure in more or less well defined tasks and activities. Computers organize computing in terms of applications, files, networks, etc. This abstraction gap becomes a big problem in situations characterized by mobility, frequent interruptions, and collaboration—situations that pervasive computing is intended to support. A new paradigm, task-level or activity-based computing, has been proposed to lessen the abstraction gap and provide a better platform for pervasive computing. In this paper, we highlight some of the open issues that remain to be addressed in this paradigm.

1 Introduction

Activity-based or Task-level computing is a new paradigm for computing that tries to provide computational support for humans at a familiar level of abstraction; namely in terms of the tasks and activities the human is involved in. This is in contrast to the prevailing desktop computing paradigm that defines computing in terms of files, documents and applications, not tasks. To illustrate the difference, you may ask researcher what he is doing, and he may answer that he is writing a position paper for a workshop. However, if you look at the process from the computer's point of view, he is running a couple of acroread applications showing some PDF documents, an Emacs editor, and a shell with a history of LaTeXcommands. The computer has no notion of the "writing paper" activity. Hence, there is a gap between the abstraction level of the human and the abstraction level supported by the computing system. The result is that much time is used on opening and closing applications, on navigating user interfaces, and on reestablishing working context, when the user shifts from one task to another. In a pervasive computing environment, characterized by nomadic work, ad-hoc collaboration, and use of multiple computing devices, this is simply infeasible.

Over the years a number of systems have been designed and described that goes some way in narrowing this abstraction gap. ROOMS [5] allows the user to switch swiftly between a number of virtual desktops; each desktop was then ideal to compose the set of applications for a particular task. The SunRay system [10] introduced mobility on top of this theme. However, these systems do not tackle the basic abstraction gap;

the computing system still has no clue to the human activity, it is only the graphical user interface that goes some way in supporting activity classification.

Several research groups have focused on direct computational support for the human level. Project Aura [6, 9] introduces Task-level computing and outlines an architecture that allows computing context to migrate between devices. The ABC: Activity Based Computing project [4, 1] have designed and implemented a prototype architecture and limited end user functionality for healthcare activities, aimed at hospital clinicians. Clinicians from Danish hospitals have evaluated the proposals during a number of workshops.

ABC defines a middleware system and a programmers interface that manages activities. Activities and Activity-Based Computing are defined as:

> **Activity:** An abstract, but comprehensive, description of the run-time state of a set of computational services. An activity is collaborative, i.e. it has a number of participants.
>
> **Activity-Based Computing:** A computing infrastructure, which support suspending and resuming activities across heterogeneous execution environments and supports activity-based collaboration

In activity-based computing, however, a lot of open issues still remain. This position paper forwards a number of issues we have identified in our work with ABC, and discuss how related task-based systems, like Aura, handles these issues. We acknowledge that our primary inspiration has been case studies within the health care domain; however, we believe that the observations made are valid inspiration for all platforms that support activity based computing.

2 Computational Activities versus Human Intent

The core conceptual challenge in activity-based computing is the issue of *intent* – if a computer system is to support human activity, should the system have an understanding of the human intent or purpose of this activity? And what does the word 'understand' implie in this context?

2.1 Capturing User Intent

Ideally, computational activities reflect user intent: a "prescribe medicine for patient Hansen" activity will, when resumed on any available device, reestablish the complete computational working context associated with this tasks, that is: start applications, find data for patient, show proper views, ect. Thus, the physician can just continue working on prescribing medicine for Hansen. The question arises, however, who defines the intent—and when and how? Sousa et al. [9, §2.3] states that given a more sophisticated context awareness monitoring *the less Prism has to rely on explicit indications from a user concerning their intentions.* There are actually two points made in this statement. First, users may explicitly state intentions. Second, context awareness can provide all or part of the user's intention.

We have discussed the first point with clinicians at our workshops and find it problematic if users are to give lengthy descriptions whenever they define a new activity. Even the relatively simple action of *naming* an activity when it is created is often too distracting. An extreme scenario is the physician that is called for immediate help with a patient suffering heart-attack, rushing to the nearest computer to get vital information only to be met by a friendly dialog-box: "Please enter a name for the activity". While this is of course an extreme situation, it does show that the problem is not trivial. Experiments with default names helped a little, esepcially when naming according to contextual information, like the patient nearby. But it did not address the fundamental problem of revealing intent to the computer system – defining a name is only a clue for humans as to what the intention is.

This leads to the second point, namely that context awareness can define user intention. We have experimented with contextual triggering of activities [3]. Our experience is that it is very difficult to correctly infer a satisfactory complete set of attributes to define user intent uniquely from environment monitoring and context awareness—even in what first appears as obvious cases. As an example, one of our workshops had the theme: prescription of medicine. We initially envisioned that some environmental triggers, like location, time, or nearness of a set of artefacts, could define that a physician is most likely to want to initiate a "prescribe medicine for patient X" activity. However, in practice physicians prescribe medicine at odd hours and in all sorts of places. Our conclusion was that a medicine prescription activity is triggered by sources too complex to capture except by explicit indication by the user.

2.2 Intent or Light-weight Activities

This leads to another open issue: Humans have intentions with their tasks. Do computational activities have intent that must be modelled in order for them to be useful? Sousa et al. [9, §4] also raises the issue: *. . . Aura should prove useful even with no deeper knowledge of the task . . .*

A recurrent debate at our workshops with clinicians was the question on whether activities should mirror human tasks. On one hand, if the computing infrastructure has accurate knowledge of defined tasks, then transferring tasks from one clinician to another could be supported directly by the infrastructure. This happens all the time in a hospital, during the shifts when for instance the evening nurse takes over from the day nurse. Thus instead of talking to the next nurse on guard (potentially forgetting some important tasks), the nurse leaving could simply transfer the list of pending tasks to the next nurse.

We found a lot of problems with this seemingly appealing idea. First, our clinicians did not like this idea. Human communication is important, and speech is much faster than keyboard exercises on the computer. Second, our activity infrastructure drifted towards a workflow system. Workflow systems dictate how work is done; we in contrast wants to support current work practices where interruptions are not an evil but key tools for communication and learning.

The current approach is to let a computational activity snapshot and reestablish composite state information in a set of applications. The link to the intent of the activity is left for the users, or their organizational setting, to establish. We hence use the term

“light-weight”about the activity because there is no explicit link between a computational activity and a human intent. The infrastructure supports fast activity switching, seamless migration to new devices, and ability to transfer computational activities to other persons as *part of* handing over responsibility and *not* as the mechanism to do so.

2.3 Activity Lifecycle

Another important issue is a activity’s life-cycle. Both ABC and Aura view activities as disjoint entities where one activity is “alive”/resumed and all others are paused. To humans, however, activities are not disjoint but more characterized as a fluid continuum. Consider a nurse’s activity pouring medicine for a patient into a medicine tray. This activity computationally embody the electronic patient record service running on the proper patient and having views on the medicine schema showing medicine prescribed and perhaps a web browser showing the medicine handbook. Next, the nurse needs to bring the medicine tray to the patient and document that the patient has indeed taken the prescribed medicine. The latter may logically seem another activity; but the nurse may view it as one activity fluidly turning into the next. From the computation point of view, it is properly also the same information and services that are needed. Maybe the light-weight interpretation of activities, outlined in the previous section, is more in line with the fluid way humans perceive their activities.

However, it poses a problem with how to denote and label activities. We need some kind of handle to allow us to manipulate the set of activities that we have presently paused in our computing system. If we label activities with text strings to ease browsing (as done in ABC), then what is the proper label for such activities that fluidly alters contents and intention? Text labels seem like a poor choice, but the problem is what the replacement is?

2.4 Organizing and Managing Activities

A physician or nurse is handling a large set of patients. For each patient, there are a number of tasks to perform. Thus, we can readily envision that each user must manage a large number of computational activities, and the question arises how to support management and browsing in a way that does not overwhelm the user. At our workshops, we have experimented with both hierarchical and linear organization of activities allowing them to be browsed. However, as stated earlier, the limited scope of our workshop did not allow us to assess browsing problems. Here the work done on user-interfaces to activity-based computing system are of interest. The Kimura system [8] provides mechanisms to organize your work in related tasks and to organize them on large wall-sized display. You then have a peripheral awareness of what is going on in the activities which ’runs in the backgound’. The WORKSpace project [2] similarly have ways of organizing related work in a ’workspace’ which can be shared with others.

2.5 Automation in Pervasive Computing

An interesting issue often raised is the ability for a computing system to react intelligently on context and execute tasks on behalf of the user without asking. In a task oriented system, human activities may then be inferred from context information. Project

Aura discusses a scenario where Aura is taking action based on inferred activity: Fred is rushing to a conference room, thus Aura downloads files and software to the computer in the conference room and turns the projector on.

At one of the first workshops, we presented our vision of helping nurses to *automatically* record medicine given to patients in the electronic patient record simply based on location tracking of medicine trays, nurses, and patients. We found it appealing that this was done simply when placing the medicine tray on the patient's bed table. However, the clinicians were very upset with this idea. First of all, they pointed out that medical decisions were now made by the computing system, not by trained, clinical, staff. Second, the actual action to take is not uniquely defined by the external triggers we kept track of, such as time schedules, and location of people and things. Indeed, complex human decision making based upon knowledge of the patient and the situation. Third, even the same context triggers only define a space for user intent, not an exact point. Even given the pretty comprehensive context triggers: Nurse Berg is near patient Anderson; the 12 o'clock medicine for Anderson is not yet given; it is around 12 o'clock; Anderson is in is her bed; the medicine tray of Anderson is on the bed table near Anderson; we could not infer whether Berg wants to document that all medicine has been given to Anderson or just some of it (unless we track each individual tablet!). We can infer that nurse Berg probably wants to do something about medicine recording for Anderson but not precisely.

Our proposal was thus to 1) infer a set of likely user intents 2) define activities for each in the set and 3) present these suggested activities *non-intrusively* for the user (see also [3]). We concluded that 'automation' in term of higher-level activities should be done with causion and suggestive. We are currently investigate automation for more low-level technical issues, like adaptation to changes in network or execution environment.

3 Programming Activity-Enabled Applications

From an architectural and software engineering perspective one of the central issues in activity-based computing is, what is required by programmers in order to write services for such a task level/activity based computing infrastructures.

Aura uses XML formats to capture activity state and services/applications are responsible for creating and parsing such descriptions during when a task is suspended and resumed. A similar approach is taken in ABC by storing state information as Java objects as key/value pairs. Helper classes and UI-wrappers that automatically does state handling exists in the ABC Framework, but in principle creating and parsing application-specific state information is the responsibility of the applications.

The downside is that state check-pointing and parsing is entirely left to the programmer, which also means that access to the source-code is necessary. Hence, it is difficult to use existing application in an activity-based computing infrastructure. Plug-ins, macros, and other kind of code need to be written for MS Word, Emacs, and other application that participate in an activity. Suggestion to use Virtual Machines (like the Java VM) to support weak or strong migration have been suggested [7]. Our experience so far is, however, that we want the state information to be (i) very light-weight because

we use it in synchronous collaborative application sharing, and (ii) more abstract than migrating code, because we want to be able to migrate between highly heterogeneous devices, like a wall-display and a mobile phone. Clearly a Java VM exists on both platforms, but the execution environment is so different that it makes little sense to migrate running code. Hence, there is a need for a simple, small, and abstract state description.

We therefore think that it is important to find techniques that relieve the burden on a programmer to program generic state handling in an ABC-service. For instance, remote method invocations schemes allow the programmer to define just an interface, and tools create stubs, skeletons, marshalling, and network code automatically. We are currently addressing this problem. An initial idea is to annotate relevant object attributes as 'state' attributes that must be part of an activity's description. This meta information may the be used actively by the activity based computing infrastructure at run-time. Another challenge is to find techniques and methods for making activity-based computing support 'application transparent', i.e. that an application can participate in activity-based computing without changing it programmatically.

4 Conclusion

Activity based computing has the promise of better supporting humans working in context characterized by mobility, frequent change of what computing device to use, and frequent interruptions. As such, it presents itself as an interesting paradigm on which to base pervasive computing systems. However, the ideas and proposals are still very new, and there are a lot of questions that are still open. We have outlined some of these, and hope to generate interest and discussion at the workshop, and hopefully get some insight and new inspiration.

References

1. ABC: Activity Based Computing. www.pervasive.dk/abc.
2. M. Büscher, M. Christensen, P. Mogensen, D. Shapiro, and P. Ørbæk. Creativity, Complexity, and Precision: Information Visualization for Landscape Architecture. In *Proceeding of ACM International Symposium on Information Visualization*, pages 167–171, Salt Lake City, USA, Oct. 2000.
3. H. B. Christensen. Using Logic Programming to Detect Activities in Pervasive Healthcare. In *18th International Conference on Logic Programming ICLP 2002*, pages 421–436, Copenhagen, Denmark, Aug. 2002. LNCS 2401, Springer Verlag.
4. H. B. Christensen and J. E. Bardram. Supporting Human Activities — Exploring Activity-Centered Computing. In *Proceedings of Fourth International Conference on Ubiquitous Computing, UbiComp 2002*, Göteborg, Sweden, 2002.
5. J. D. Austin Henderson. Rooms: the use of multiple virtual workspaces to reduce space contention in a window-based graphical user interface. *ACM Transactions on Graphics*, 5(3):211–243, July 1986.
6. D. Garlan, D. Siewiorek, A. Smailagic, and P. Steenkiste. Project Aura: Towards Distraction-Free Pervasive Computing. *IEEE Pervasive Computing*, 1(2):22–31, 2002.
7. M. Kozuch and M. Satyanarayanan. Internet Suspend/Resume. In *Proceeding of the Fourth IEEE Workshop on Mobile Computing Systems and Applications (WMCSA'02)*, pages 1–7, 2002.

8. B. MacIntyre, E. D. Mynatt, S. Vodia, K. M. Hansen, J. Tullio, and G. M. Support for Multitasking and Background Awareness Using Interactive Peripheral Displays. In *Proceeding of ACM User Interface Software and Technology 2001 (UIST01)*, pages 11–14, Orlando, Florida, USA, Nov. 2001.
9. J. P. Sousa and D. Garlan. Aura: an Architectural Framework for User Mobility in Ubiquitous Computing Environments. In *WICSA*, 2002.
10. Using Smart Cards With the Sun Ray 1 Enterprise Appliance. Whitepaper from Sun MicroSystems, available from www.sun.com, Sept. 1999.

About the Authors

Jakob E. Bardram's main research areas are pervasive computing, distributed system, software architecture, and computer supported cooperative work (CSCW). His main focus currently is 'Pervasive Healthcare' and is conducting research into technologies of future health - both at hospitals and in the patient's home. He is a principal architect of the Activity-Based Computing (ABC) Framework which includes support for activity migration and shared activity collaboration, among other things. His current interests include software architectures and frameworks for activity-based computing and collaboration, context-awareness, and distributed computing in a heterogeneous, pervasive computing environment.

Henrik B. Christensen's research interests include software architecture, software engineering, pervasive computing, and teaching. As researcher at Centre for Pervasive Healthcare, he has been architect on the ABC framework that experiments with computer support for human activities and collaboration. His research includes conceptual frameworks, architectures, and programmers API's for activity based computing, proactive discovery of activities, and techniques and tools for engineering of reliable pervasive services.