Adaptable Pervasive Flows – An Emerging Technology for Pervasive Adaptation

Klaus Herrmann, Kurt Rothermel
University of Stuttgart, Germany

Gerd Kortuem
Lancaster University, UK

Naranker Dulay
Imperial Collage, London, UK

Abstract

The era of pervasive computing brings with it a grand challenge: Pervasive applications must adapt to the dynamics entailed in human behavior and constantly changing computing environments. In this paper, we propose Adaptable Pervasive Flows as a novel technology that goes far beyond existing approaches for adapting pervasive computing systems. APFs model applications in a fashion similar to classical workflows while being situated in the real world. The notable advantage of this is that applications as well as their environment can be adapted proactively based on knowledge about future tasks. We introduce the visions, concepts, and challenges of this emerging approach.

1 Introduction

Humans are increasingly embedded in an environment consisting of growing numbers of computing devices and artifacts that provide various degrees of computing power and awareness. Current estimations are that ten years from now, there will be 7 billion people surrounded by 7 trillion wireless devices and sensors living on this planet[1]. Pervasive applications are software systems that run in such environments in a massively distributed fashion and support mobile human users in their daily activities. They shall provide adequate computing and communication services in an *anywhere-and-anytime* fashion transparently to the user.

This vision holds several scientific challenges ranging from adequate hardware architectures to communication protocols and aspects of software distribution. The most challenging question, however, is:

> *How can a pervasive application adapt to the user in order to support him/her in an unobtrusive way?*

Ideally, the application runs in the background, unnoticed by the user, and adapts to his actions. This requires new paradigms for programming such applications and their interactions with the users. Today, these interactions are still done explicitly and manually in most cases which contradicts the idea of unobtrusiveness. Novel methodologies and technologies are needed to adapt applications to a changing environment and to dynamic user behavior.

In this paper, we present the vision and the concepts of *Adaptable Pervasive Flows* (APFs, also simply called *flows* hereafter), a methodology for engineering pervasive applications that are able to adapt themselves and the human user's environment to his goals and activities. Our research on flows is conducted in a European project called *ALLOW*[2] that is funded under the 7^{th} Framework Programme. In this project, we take a broad approach to the problem of pervasive adaptation penetrating all the relevant problems.

Adaptable Pervasive Flows apply concepts similar to classical workflows as a basis for adaptation. Many processes in real life are defined in terms of flows, either implicitly or explicitly. A flow is a computer-based model that essentially consists of a set of *actions*, *glued together* by a *plan* (or control flow) which defines how the actions should be performed to achieve some *goal* under a set of *constraints*. Flows are explicitly tailored (1) to being executed in pervasive environments, and (2) to being adaptable. They are *situated in the real world*, i.e., they are logically attached to entities like artifacts or people, moving with them through different contexts. While they are carried along, they model the behavior intended for the associated entity, and *adapt the entity's environment to this behavior*. Thus, when a mobile user carries a flow that specifies his prospective actions, the pervasive computing machinery in his environment will be set up for him by the flow. Since people may change their minds, and since artifacts and people may be subject to changes in the environment, *the flow itself may also adapt* to reflect such changes. This requires flows to be *context-aware*. They can take into account the context pertaining to their entity's current environment as well as the entity's actual activities in order to dynamically adapt to changing situations.

The paper is structured as follows. In Section 2, we give a conceptual overview of adaptation in pervasive systems.

[1] Source: Wireless World Research Forum

[2] This research has been supported by EU-FET project 213339 ALLOW.

Published in In Proceedings of the 2008 Second IEEE International Conference on Self-Adaptive and Self-Organizing Systems Workshops (SASOW 2008).

http://dx.doi.org/10.1109/SASOW.2008.25

Section 3 presents the related work in this area. The main concepts of the flow technology are investigated in Section 4. Conclusion and future work are discussed in Section 5.

2 Adaptation in Pervasive Systems

Employing the anytime-anywhere metaphor of pervasive computing to provide support to mobile users in their real-world tasks has an important implication: Applications that run in such environments are subject to high degrees of dynamics. This is the result of intermittent connectivity, varying availability of resources (devices, services, etc.), changing quality of service, and other unpredictable variations that occur as mobile users move through different environments and contexts.

On the other hand, users expect a stable, robust, and predictable behavior from an application. Otherwise, the usability of the application and the user's confidence are drastically decreased, which means that most users would refrain from using the application.

This fundamental conflict can only be resolved if the pervasive application and the supporting system software are able to adapt to relevant changes. But what exactly does adaptation mean? In the following, we will answer this question for pervasive computing in general terms. We define, what adaptation actually means in such environments and what the design space for pervasive adaptation is.

2.1 The Purpose of Adaptation

The purpose of adaptation is to adjust a system to match its environment despite the fact that this environment is constantly changing. On the other hand, a system should adapt to preserve certain invariant properties (perceived by the user or other systems) over time in the face of external changes that can cause these properties to deviate. Zadeh [8] defines that a system is adaptive if it *performs acceptably well* under all practically occurring input functions. Here, "acceptable performance" is the general invariant, and "performance" can relate to any measurable behavior of the adaptive system. Preserving system invariants by adapting is the measure taken to make the system appear stable and predictable from the user's point of view.

2.2 The Subjects of Adaptation

There are essentially three subjects that we may need to adapt in order to preserve invariant system properties:

The pervasive application: The core application itself and its components.

The environment: Those hardware and software components that do not directly belong to the pervasive application but that influence its behavior.

The user: If the system is not able to apply appropriate adaptations, it should either inform the user that application performance is about to deteriorate, or make suggestions indicating a behavior that would lead to the preservation of invariants (e.g. move more slowly).

2.3 The Triggers of Adaptation

Changes that make adaptations necessary usually come from two different sources:

The environment: Pervasive applications are inherently depending on resources provided by the local environment (devices or services). When the availability or quality of these resources changes, the application may experience deviations from its invariants due to emerging mismatches in resource supply and demand [6].

The user: As the user moves through the world and acts, his behavior may violate the current assumptions of the application and cause deviations from its target invariants. Thus, adaptations are triggered to follow his actions in the world and make the application fit them.

2.4 The Entities Executing Adaptations

Application developers should not be faced with designing and developing the basic mechanisms of adaptation. This shall be hidden as much as possible inside a special software component that is responsible for deciding about and enforcing adaptations across different applications and environments. We call this software components the *adaptation middleware*. The task of this adaptation middleware is usually to establish and monitor the bindings of the application to the necessary resources in the environment. Furthermore, it must provide ways of specifying the invariants desired for applications and mechanisms for preserving them by adapting. This should be as transparent as possible for the application, the developer, and the end user.

However, it may not be possible to encapsulate every adaptation inside the adaptation middleware. The application itself may have to be designed in a way that allows for highly application-specific adaptations. The adaptation middleware should also provide means to application developers to achieve this in a standard way if necessary, avoiding proprietary application-side solutions.

2.5 The Timing of Adaptation

There are essentially two dimensions to the timing of adaptation. The first dimension defines the time at which

the adaptation is executed relative to the triggering event: An adaptation can be executed *reactively* (in reaction to a triggering event that has already occurred), or *proactively* (before the triggering event actually occurs). Proactive adaptations are most desirable as they can prevent failures and instabilities rather than trying to cure them. This ensures that the user experiences disruptions less often. However, proactive adaptations are also much more challenging as they require information about future events.

The second dimension in the timing of adaptations defines the time scales over which adaptations are executed. At one extreme end, there are *short-term adaptations* that affect a specific running instance of an application. The next instance starts from the same initial state and undergoes new adaptations. At the the other extreme end, there are *long-term adaptations*. In systems that exhibit long-term adaptation, the adaptations applied to each application instance are used to learn from them and to improve the application with every run. Each new application instance builds on the instances before it and benefits from the learning process. For example, if a certain resource R_1 tends to break down and has to be replaced by R_2 in every application instance, the system may learn from this and decide to use R_2 from the very start as this saves time and effort.

3 State of the Art

There are a number of different approaches for adaptation in pervasive systems. Most systems deal with the management of resources under mobility-induced dynamics. For example, Gaia [5] enables the easy access to resources in a habitat by employing an extension of the model-view-controller paradigm. Sessions are used to allow mobile users to move across different locations and have their data and applications available. PCOM [1] allows for adaptation by using a component-based approach. Applications are composed from components based on contracts and re-configured at runtime when components become unavailable due to changes. These systems may adapt an application reactively to changes in the environment.

Aura [3, 7] and activity-oriented computing [2] extend these approaches by explicitly modeling user tasks and intentions as a basis of adaptation. These systems may also adapt the user's environment due to the knowledge of his intentions. However, there is no notion of an overall process represented by a series of tasks. Plan-driven ubiquitous computing [4] extends upon Aura and related systems by introducing a plan-based representation of user tasks to enable proactive adaptations. This approach is similar to the one introduced in this paper. However, Adaptable Pervasive Flows add a number of key ingredients such as flow situatedness, flow evolution, adaptive flow distribution, interaction between flows, etc. Therefore, they are a more rigorous and general concept that goes far beyond any existing approach.

4 Adaptable Pervasive Flows

Flows are closely related to classical workflows. They mainly consist of *tasks* being connected by *context-aware transitions*, and they model the actions of a *flow entity* over the course of a more complex overarching activity. An example is depicted in Figure 1.

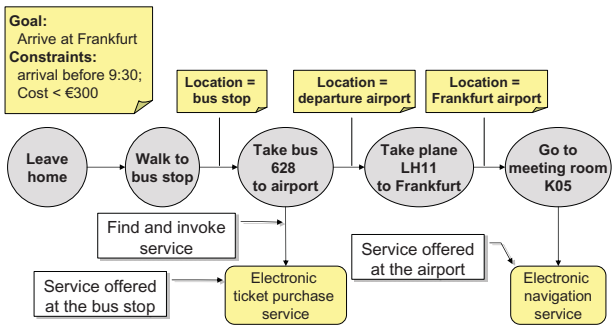

Figure 1. Flow Example

A *flow entity* (also simply called *entity*) may be a human user or some inanimate object (e.g. a container in a logistics process) on behalf of which a flow is executed. The system that is responsible for executing flows is simply called the *flow system*. This system is usually distributed and resides in the environment of the entity. The flow system is responsible for executing a flow in parallel to the real-world actions of its entity. The fundamental idea of the flow approach is that this synchronized execution allows the flow system to detect deviations of the real-world actions from the planned actions and use these deviations as the basis for adequate adaptations. In order to achieve this, we employ activity sensing technologies to recognize user activities and other sources of context information (e.g. location).

The flow in Figure 1 models a simple journey of a business traveler. Based on location information, the flow advances through its tasks, and each task can invoke supportive services in the local environment to guide the user through the whole process (see bottom of Figure 1). A very important element of the flow is a set of meta data expressing the *goals* and *constraints* of the flow. In our example, the goal is to arrive at the airport in Frankfurt, and the constraints define the latest arrival time and a maximum cost. Having this data is vital for doing flow adaptations as they define the *invariants* (cf. Section 2.1) that shall be preserved by adaptations and, consequently, imply a subset of all possible adaptations in a particular situation. The traveler's flow is attached to him and, thus, follows him through the

different stages, locations, and contexts of the process (his journey).

4.1 Flow Adaptation

A flow can be subject to *short-term adaptation* (also simply called "adaptations") and *long-term adaptations* (called "evolution") (cf. Section 2.5). A key innovation of the flow technology is that a flow represents information about the *past, current, and (prospective) future* activities of its entity. Having information about the future is obviously a huge advantage for adaptability as it allows for proactiveness. Figure 2 shows the range of adaptations possible in flow systems. On the top left of the figure, the initial journey flow (Figure 1) is adapted because the user has missed the bus. Here, we assume that the bus station is equipped with a flow system that knows possible adaptations for such situations. It has certain local knowledge (e.g. where the next taxi stand is located) and can use this knowledge to apply an adaptation that preserves the goals and constraints of the flow. This means that flow adaptation is local (and, therefore, scalable) in nature: As a flow moves through different contexts, it can be adapted in each context locally without requiring access to some centralized adaptation logic.

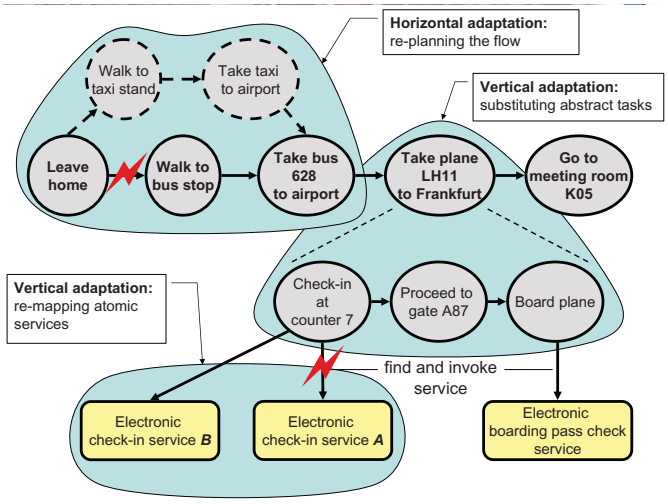

Figure 2. Forms of flow adaptation (Top left: Horizontal adaptation by re-planning the flow; Right side: Vertical adaptation by subflow substitution; Lower left: Vertical adaptation by service re-mapping)

The adaptation mechanism tries to adapt the flow such that the goals remain achievable and the constraints are met. Unlike other systems for pervasive adaptation, flows represent a structured models of an application. Therefore, they provide many potential ways of adaptation like inserting or replacing tasks or subflows, removing parts of the flow, changing transitions and their conditions, etc.

We differentiate between *horizontal* and *vertical flow adaptation*. A flow is adapted horizontally if it is re-planned. That is, if new tasks are inserted, existing tasks are removed, and transitions between tasks are changed. Figure 2 represents an example of this form of adaptation (top left). Vertical adaptation is about changing the mapping of flow tasks to resources in the environment. Tasks can either be mapped to atomic services (bottom of Figure 2), or they can be mapped to *subflows* (middle of Figure 2). In the latter case, the task is abstract and involves a number of steps (tasks) that are not known at design-time but only when the flow is actually executed in the right environment. For example, the "Take plane..." task is actually a bit more complex and requires several steps. A concrete representation of this task as a subflow is present at the concrete airport and is substituted for the abstract task as the user enters the airport. This form of late binding allows for very flexible flow design. If a task is mapped to an atomic service, vertical adaptation means that the task has to be re-mapped (e.g. because the old service became unavailable).

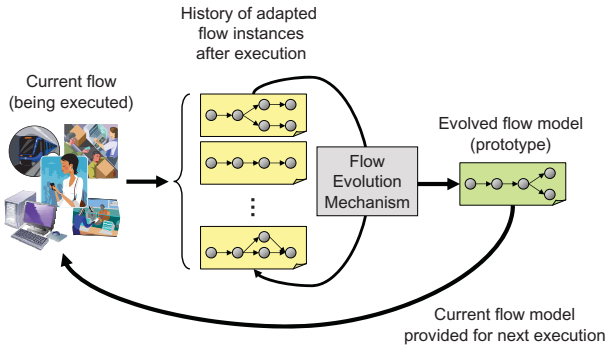

Figure 3. Flow Evolution

After the flow has finished, it represents the best achievable fit under the circumstances prevalent at the time of its execution. Such flows are not discarded. Instead, they are collected and used for evolving a common flow model from which future flow instances for the application are instantiated (see Figure 3). Evolutions are based on the same manipulations of flows stated above for adaptations. A flow's performance (e.g. runtime and ability to achieve its goals and meet its constraints) may be monitored and used to compare the fitness of several flow instances. Diverse techniques from the domain of machine learning can be used to evolve a flow model from a set of flow instances based on this data.

4.2 Situatedness

Unlike classical workflows, Adaptable Pervasive Flows are *situated in the real world*, i.e. they are *attached* to entities like artifacts or people, moving with them through different contexts. This attachment is logical in nature, i.e. in principle the location where the flow is executed and stored is independent of the current physical location of its entity. A flow could be executed in some server while the actions of its entity (e.g. a user) are captured locally and communicated to the flow. On the other hand, the flow could also reside on the mobile device of its user. Logical attachment provides means for maintaining a relation and decoupling at the same time, and it avoids the need for users to carry computing devices. All that is necessary is a way of identifying an entity (e.g. RFID).

4.3 Flow Distribution and Mobility

In many cases, there is an inherent need for executing a flow in close proximity to its user[3]. This is mainly due to the interactions that need to be conducted between the user and the flow. Therefore, it can be beneficial to execute the flow in the current environment of the user in order to colocate it with the resources it uses (e.g. services the tasks are mapped to). However, considering the wide range of possible environments, it may well be that there are only small-scale low-performance devices available near-by. For this reason, the flow modeling language allows for flows to be fragmented and the fragments to be placed on different devices. In this way, flow execution is distributed. Scopes are used to specify relations between the tasks of the flow, and these relations may be used to control the distribution. E.g. tasks that need to communicate heavily may better be put on the same device in environments with a low-bandwidth network. The distribution of flows must also be adaptable to changes in the properties of the environment and user mobility. Flow distribution is controlled by a utility function that regards aspects like communication bandwidth, communication cost between flow fragments and between tasks and services, the availability of devices, etc.

4.4 Context and Activity Sensing

Employing flows for adaptation requires feedback from the real world in oder to be able to detect deviations between the real-world events and the flow's plan. This feedback includes context information like time, location, the status of relevant entities in the environment etc. However, the core objective of Adaptable Pervasive Flows is to achieve human-orientation. That is, a flow-based system should first and foremost be able to adapt to the actual needs of its human users. Therefore, a special kind of context information that is employed in the system is *activity information* about the current real-world actions executed by humans. By collecting raw sensor data (e.g. from accelerometers) and recognizing activities from this data, the flow system can assess whether a human user is actually doing what the flow prescribes or whether the flow must adapt to the user. If, for example, our business traveler should be running to catch the plane but instead, he is walking slowly, the system could inform him about the situation and tell him to speed up, or it could proactively adapt his environment as it is shown in Figure 4.

[3]For simplicity, we assume here that the flow is attached to a human user. However, similar observations can be made for flows that are attached to inanimate objects.

4.5 Environment Adaptation

Many processes in real life are associated with adapting (setting up, configuring) certain technical equipment in the environment. In flow-based systems, this is done by the flow driving the overall process. In order to allow for this kind of adaptation, the equipment exports services that enable flows to issue adaptation requests. Apart from technical equipment, adaptations may also pertain to humans that are directly or indirectly involved in the process.

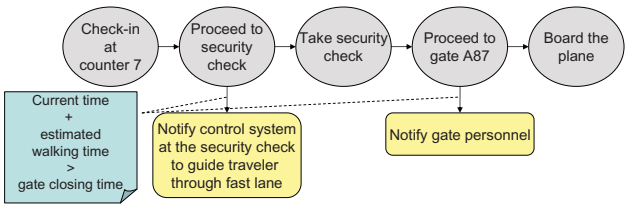

Figure 4. Environment adaptation example

An example is presented in Figure 4. Here, we assume that the traveler is in a hurry and trying to catch the plane. This can be inferred from the current time and parameters that are available in the environment (estimated walking time to the gate and gate closing time). As the flow system recognizes that the traveler may not be able to make it in time, it adapts the security check system to guide the traveler through the fast lane. Additionally, it notifies the gate personnel that the traveler is on his way, such that the personnel may wait longer. In this case, the electronic security check system as well as the personnel at the gate are considered to be belonging to the environment and different adaptations are issued to ensure the success of the traveler's flow.

4.6 Adaptive User Interfaces

For the interaction of the user and his flow, adequate user interfaces are required. Interacting with a user through his mobile device is an option if the user is actually carrying such a device. However, we explicitly foresee, applications where this is not the case and in which other means of interaction are required. For this purpose, the concept of *interaction spheres* is introduced. An interaction sphere defines a physical space around a user's current position. I/O devices (monitors, speakers, terminals, etc.) that reside in this sphere may potentially be used for interactions. E.g. text output may be directed to a monitor in close vicinity of the user, and for text input, the user may be directed to an information terminal. Identifying and using those devices close to the user according to their properties and capabilities in a natural way is the challenge with respect to this. Note that with activity sensing, we have another way for the user of directing input to the flow system: A user may use deliberate gestures (e.g. moving hands) to create input.

4.7 Security, Trust, and Privacy

Any pervasive system that penetrates a human's life will only be acceptable if it is sufficiently secure and preserves the user's privacy. Doing this is a tough challenge in the context of this project due to the pervasiveness and the dynamics of the whole system. A user's private data must not be disclosed to unauthorized parties. His transactions must be secured to avoid fraud. To achieve this, we plan to use reputation systems that are capable of assessing the trustability of an entity based on statistical knowledge about its prior actions. Such data about an entity A can be captured as the system is running and provided to other parties that are not required to know A before hand. Thus, statically configured access rights can be avoided and the system may also adapt its level of trust based on what users are doing.

5 Conclusions and Future Work

The new programming paradigm for adaptable pervasive applications based on the concept of flows holds a large potential. It promises to enable truly proactive adaptations to dynamic changes in the environment and the application due to the fact that a flow represents an explicit model of the future actions within an application. This paves the way towards human-oriented pervasive systems that adapt in an unobtrusive and seamless way. We have presented the broad range of concepts tackled in this international research effort to realize this vision. Many of them are very challenging, but we believe that a successful coherent effort to solve them can result in a major breakthrough.

The research on flow-based systems is still at its very beginning. Over the next three years, we will follow a rigorously defined agenda that ranges from basic models for flows, context, adaptation, evolution, distribution, user interfaces, and security to fully functional demonstrators in order to validate our concepts.

References

[1] C. Becker, M. Handte, G. Schiele, and K. Rothermel. PCOM - A Component System for Pervasive Computing. In *Proceedings of the Second IEEE International Conference on Pervasive Computing and Communications*, 2004.

[2] 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*, 2002.

[3] D. Garlan, D. Siewiorek, A. Smailagic, and P. Steenkiste. Project Aura: Toward Distraction-Free Pervasive Computing. *IEEE Pervasive Computing*, April–June 2002, 2002.

[4] G. Look and S. Peters. Plan-driven ubiquitous computing. In *Student Oxygen Workshop Cambridge*. MA: MIT Project Oxygen, 2003.

[5] G.-C. Roman, J. Payton, R. Handorean, and C. Julien. Rapid Deployment of Coordination Middleware Supporting Ad Hoc Mobile Systems. Technical Report WUCSE-2002-4, Washington University, Department of Computer Science, St. Louis, Missouri, Mar. 2002.

[6] M. Satyanarayanan. The Many Faces of Adaptation. *IEEE Pervasive Computing*, July-September:4–5, 2004.

[7] J. Sousa, V. Poladian, D. Garlan, B. Schmerl, and M. Shaw. Task-based adaptation for ubiquitous computing. *IEEE Transactions on Systems, Man, and Cybernetics, Part C: Applications and Reviews*, 36(3):328–340, 2006.

[8] L. Zadeh. On the Definition of Adaptivity. *Proceedings of the IEEE*, 51:469, Mar. 1963.