Adaptability and Interfaces: Key to Efficient Pervasive Computing

Arkady Zaslavsky
School of Computer Science and Software Engineering, Monash University
Melbourne, 900 Dandenong Rd, Caulfield East, VIC 3145 AUSTRALIA
a.zaslavsky@monash.edu.au

As mobile and embedded computing devices become more pervasive, the nature of interaction between users and computers is evolving. Computing applications need to become increasingly autonomous and invisible, by placing greater reliance on knowledge of context in which they are running and reducing interactions with users. Moreover, due to the current popular emphasis on `anytime, anywhere' computing, applications must cope with highly dynamic environments in which resources, such as network connectivity, bandwidth, security and software services, vary over time and therefore applications' behaviour and functionality have to adapt (adjust) to currently available resources and constraints. One example of varying resources comes from a requirement that users can use multiple and potentially heterogeneous computing devices and switch between those (eg, a workstation to a mobile laptop, or PDA (Personal Digital Assistant), then to a mobile phone) while still participating in distributed computational activities. To enable such change of devices, applications have to adapt to a variety of displays, various CPU capabilities, presence or absence of a hard disk and varying bandwidth of communication channels.

Pervasive computing is believed to be a computing paradigm incorporated in a variety of devices (computers, cars, entertainment units, appliances, etc.), which can carry out computational activities in a relatively non-intrusive manner and can impact and support many aspects of work and everyday life. Pervasive systems need to be aware of their environment and associated resources (i.e. aware of their context), able to detect changes in the environment (context changes) and can adjust/adapt their functionality and behaviour to the changes. Pervasive computing systems represent a natural extension and evolution of mobile/ubiquitous/nomadic computing systems, which first emerged in the early 90s.

While recognizing diversity and enormity of pervasive computing related issues, I reckon that ultimate success of this paradigm depends on useability of pervasive applications. I would identify the following important issues that need to be addressed by mobile and pervasive computing community.

- Adaptability
- Flexible customisable interfaces, both present in the device and around it
- Context management, including location
- Accuracy & credibility vs. disconnection-transparent continuous execution, mobile transaction accuracy thresholds
- Direct and indirect methods of location (and other context attributes) verification

The rest of this position paper is structured accordingly.

Adaptability. System stability could be viewed from the perspectives of functional and behavioural stability. Functional stability refers to systems adherence to design specifications and its ability to achieve the expected outcomes from individual components as well as from the system as a whole. Behavioural stability refers to interactions between components, expected behaviour and proactive moves predominantly at run-time.

Mobile computing system is a complex system consisting of both hardware and software components. Its behaviour could be represented by a trajectory in a multidimensional space where each dimension represents a domain of acceptable values (called acceptance region) for a particular context attribute. The system's state can be represented by a point in this N-dimensional space with coordinates that take values from discreet or continuous scales. Figure 1 illustrates this representation in a simplified 3-D space with bandwidth, cache size and response time as context attributes and respective dimensions.

Proactivity of the system is determined by its ability to foresee changes, control them and remain within the acceptance regions where the system is stable. Proactivity is based on past patterns and history as well as forecasting and what-if analysis.

Stability can be measured by $\sigma = \beta S_b \div \chi S_f$ where β and χ are weighs influenced by context, while S_b and S_f are metrics-variables measuring behavioural and functional aspects of stability. Developing ways to measure and control stability is one of the important issues for future research. Adaptivity (or synonymously, adaptability) and stability are closely related, interdependent and are in a functional relationship S = f (A) where A is the measure of adaptivity of the system. General systems theory could be used for describing and analysing stable adaptable systems.

The monitoring component is an important part of the system. Continuous monitoring of the environment and the system's states determines when the adaptation will be applied. When the system's current state (that is, all the attribute variables of the current context), is within the ε-subspace, the system is considered stable. When as a result of reaction to internal and/or external changes, one or more variables fall outside the respective ε-vicinity, the adaptation mechanism will generate one or more actions/events to stabilize the system. When the trajectory segment lies within the ε-subspace, it is not necessary to apply adaptation engine except when the forward analysis detects potential future fluctuations, for instance, a possible blackout in a tunnel and consequent change into a disconnected state.

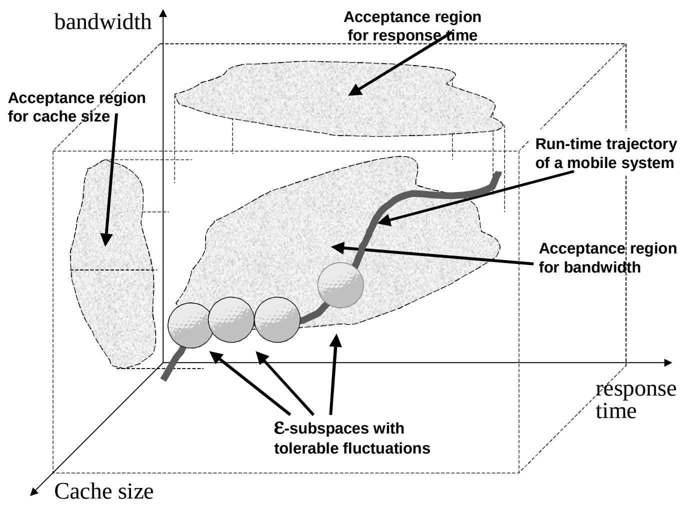
bandwidth
Acceptance region for response time
Acceptance region for cache size
Run-time trajectory of a mobile system
Acceptance region for bandwidth
response time
ε-subspaces with tolerable fluctuations
Cache size

Fig.1. Trajectory of a pervasive system.

Adaptability in pervasive systems can be divided into internal adaptability (internal adaptation of the application as a reaction to the change in the environment – such applications are often called context-aware applications), external adaptability in which the environment around the application is adapted – this adaptation can be completely transparent to the application or it could be prescribed by the application but carried out by the environment as a reaction to the change (environment dependent adaptation) or a hybrid adaptation (both internal and external). Adaptability methods must ensure that adaptations are within a functional and behavioural range required by the user, are optimal as far as system resources are concerned, and adaptations are not cascading.

Flexible customizable interfaces. Physical devices like mobile phones, PDAs, and other specialized means for external communication could be replaced by one non-intrusive and personal universal communicator (**Ucom**). This could be worn as a necklace, watch, headband or in a pocket. Ucom could be a gateway to outside network for the personal area network (PAN). Its function will be to discover and enable interaction with ambient devices including interfaces needed for a specific task. These interfaces could be a keyboard pulling out of a pay-phone, LCD newspaper that could be used as a disposable display device with touch-screen sensitivity, an embedded chip in a fan, or a disposable keyboard bought from a news agency in packs of ten. The deluxe version of the Ucom device could come with a micro-projector, accepts voice inputs, and does voice recognition (currently available on iPaqs). The Ucom could also provide a personal interface to computational resources like workstations. In the future, Ucom will be able to generate 3D hologram artefacts taking shape of an interface customizable to user's current communication needs, eg, a steering wheel, a keyboard, a joystick.

Context Management. Meaningful adaptation of a system/application is impossible without the system knowing the context in which it is running. The issue of context description is complex and still poorly understood. The main reason is that the context information needs to show both requirements and current status of various objects. The context description needs to capture user requirements/profile, application requirements (type of the environment the application can be executed on, service quality which the application expects from this environment including bandwidth, security, disconnections, etc.) and device capabilities. These requirements/capabilities define acceptance regions for context values. For each context value its ε-vicinity can be defined. Context information also needs to capture some relationships between contextual information. These relationships need to capture basic contextual relationships (e.g. on which operating systems a given application can execute) as well as very complex and not yet well understood/modelled dependencies between context descriptions. The capabilities of a self-adaptive software include: detecting a change in context or a change in need: (detection of deviations from the initial commitments or QoS contract); knowing the space of adaptation: knowledge of the choices offered to fix the deviation; reasoning about adaptation decision: revising goals by taking into account recent changes and make commitments (maybe after re-negotiation); doing the adaptation action.

The whole architecture for management of current context is usually hierarchical and its levels are respectively responsible for context gathering (from a variety of hardware and/or software sensors), context interpretation (which may involve complex processing of raw sensor data) and context description (the final context description represented in a general form readable by the system). Context sensing and interpretation methods depend on the type of context and are reasonably well understood for many types of context. Context attributes can be independent (a change in the attribute does not have any impact on other context attributes) or dependent.

Accuracy, credibility and commit/abort ratio in transaction management. A number of observations could be made with regards to transaction management models in the context of mobile databases. Firstly, if current transaction models are projected onto mobile computing environment without major changes, the number of transactions that have to be aborted increases greatly. This happens because of intermittent connectivity, timeouts, host mobility. Secondly, it is hard to preserve ACID properties without sacrificing performance and useability of applications. Thirdly, we need to differentiate between applications that require absolute accuracy and consistency of data from those applications that may suffice with accuracy within margins specified by the user or derived from history or patterns.

Direct and indirect methods of location (and other context attributes) verification. While authentication and verification of the object or a person behind the device interface is a problem for any application, be it wired or wireless, it takes on higher priority and urgency for wireless and/or mobile applications. The argument for that is the greater delegation of decision making and inherent time pressure for users who are mobile. Greater reliance on and/or proliferation of mobile agent technologies combined with granting those software agents greater autonomy and independence makes imperative the development of direct and indirect techniques that can assure users about the identity and authority of their software interlocutors. What are the guarantees that the person who participates in a video-conference is not a software impersonation ? Is it not tempting, given increased workload and responsibilities, to have an unlimited number of specialized software clones of self which could take on routine and seemingly unimportant activities delegated by the user ? How to combine these virtual and real worlds in a pervasive computing system ?

I would like to conclude by reminding that design of adaptive applications is driven by one main inspiration, to give the best service possible under the given set of conditions. This design philosophy is not new, with robust, re-configurable, dependable systems considered to be cousins of adaptive systems. The challenge in heterogeneous and mobile environments is to redefine the design of applications so that they can match the device and/or environmental constraints. Adaptive (or adaptable) systems are guided by the Ashby’s principles:

- **The law of requisite variety**. Control can only be obtained if the variety of the controller is at least as great as the situation to be controlled.
- **The variety-adaptability principle**. Systemic variety enhances stability by increasing adaptability.
- **The over-specialization principle**. Too much of a good thing may render systems unstable in the face of environmental change.
- **Darkness principle**. No system can be completely known.
- **The environment modification principle**. To survive, systems have to choose two main strategies: One is to adapt to the environment, the other is to change it.
- **The maximum power principle**. Those systems that survive in competition between alternative choices are those that develop more power inflow and use it to meet the needs of survival.
- **The feedback principle**. The result of behaviour is always scanned and its success or failure modifies future behaviour (learning from experience).

Acknowledgement. Some of the issues above have been discussed and worked on jointly with m3 (Enterprise architecture for mobile computations) team at the national Cooperative Research Centre for Distributed Systems Technology Centre (CRC DSTC Pty Ltd), Australia.