(2) Sustainable Evolutionary Systems

Katsuya Kawamura, John Hamard, Robert Hirschfeld, Atsushi Minokuchi and Bertrand Souville

DoCoMo Euro-Labs is developing sustainable evolutionary systems with high user acceptability, aiming at flexible and efficient evolution of services and platforms. We are researching technologies for reconfigurability, service adaptation, and adaptive user interfaces to meet these aims.

1. Introduction

As regards future mobile communication networks, it is envisioned that in addition to heterogeneous access networks, ubiquitous computing devices of all sizes, sensors, and embedded devices will be integrated hierarchically through ad hoc and sensor networks. Moreover, from the users' point of view, secure and seamless access to highly available and attractive services is a fundamental requirement. As plentiful services are promoted by various providers, adequate user interfaces are expected to offer access to these services.

It is desirable that various solutions are utilized optimally so that systems and services provided via future mobile communication networks can be developed, deployed, and maintained in a decentralized manner. For example, the distribution of radio and computational resources in heterogeneous access networks must be optimized. This means that it is more efficient if new and existing services share common elements. In addition, it is important to consider how users can enjoy abundant services easily. To promote decentralized development, deployment, and maintenance of systems and services, a mechanism whereby individual elements can be improved at different paces is desirable. Co-existence of different services and systems will increase, and since most changes are unexpected, it is necessary to address them when they occur.

To make these mobile communication networks a reality, the technological issues listed below must be resolved:

- Flexible software evolution must make efficient use of radio and computational resources in heterogeneous access networks.
- Individual components of services and platforms must be evolvable at a different pace while certain components are to be reused. Unexpected changes must be dealt with after deployment, possibly at runtime.
- Users must be able to use several services simultaneously or switch between them without any difficulty.

The following chapters present our new approaches to solve these issues: namely reconfigurability, service adaptation, and adaptive user interfaces.

2. Reconfigurability

In the case of a reconfigurable mobile terminal, the change of the terminal configuration (including potential software updates of new air interfaces) enables the optimum air interface to be selected under a heterogeneous access network environment and assures mobility between heterogeneous access networks. Moreover, user demands can be met flexibly by adapting to changes in the state of network load and provided services.

A reconfigurable base station adaptively selects appropriate air interfaces and relevant resources and optimizes them in order to make efficient use of heterogeneous access networks. This is also an effective technique for making simultaneous use of heterogeneous access when operators install and expand future mobile communication networks.

In our current research, as an element composing reconfigurability, a scheme for more efficient and secure renewal of software components is being developed.

When software is distributed, it is necessary to consider the scarcity of radio resources on a mobile communication network. And when software is updated, the software on multiple terminals sometimes has to be renewed at the same time. Moreover, along with software-distribution renewal, assuring security across each layer is a significant challenge.

As regards studies on these challenges, it is being assumed that mobile networks and ad hoc networks will be integrated under seamless conditions. The concept of using a local server for software distribution has been introduced [1], and efficient software distribution through ad hoc networks has been proposed. As a method for simultaneous software update on multiple terminals—including fallback when updating fails—a time-out based algorithm has been developed [2]. Furthermore, in our current research, we have set up a test environment for studying

reconfigurability and verified the implementation and operation of this algorithm. Regarding security, we developed a mechanism for secure distribution of software based on web services, and we have implemented it in the test environment.

The time-out based algorithm for simultaneous renewal of software described above is shown schematically in **Figure 1**. This shows how the algorithm solves the issue that occurs during reconfiguration time or when nodes with different protocols cannot mutually communicate. This example covers modifications to the routing protocols of an ad hoc network. Under the circumstances in which new software has been distributed to multiple mobile terminals and the reconfiguration limitation time (t) has been negotiated, the algorithm shown in the flow chart in Fig.1 is executed on each mobile terminal. Under such scenarios, the assurance of software compliance between multiple mobile terminals has been confirmed [2].

In the next phase of our reconfigurability study, we will expand the mechanism for software distribution and upgrading developed up till now, and link it to the service adaptation study described in the following chapter.

With the goal of realizing future mobile networks capable of reconfigurability in mind, DoCoMo Euro-Labs—and with the aim of contributing to the definition of the requirements and overall architecture of those networks—is involved in collabo-

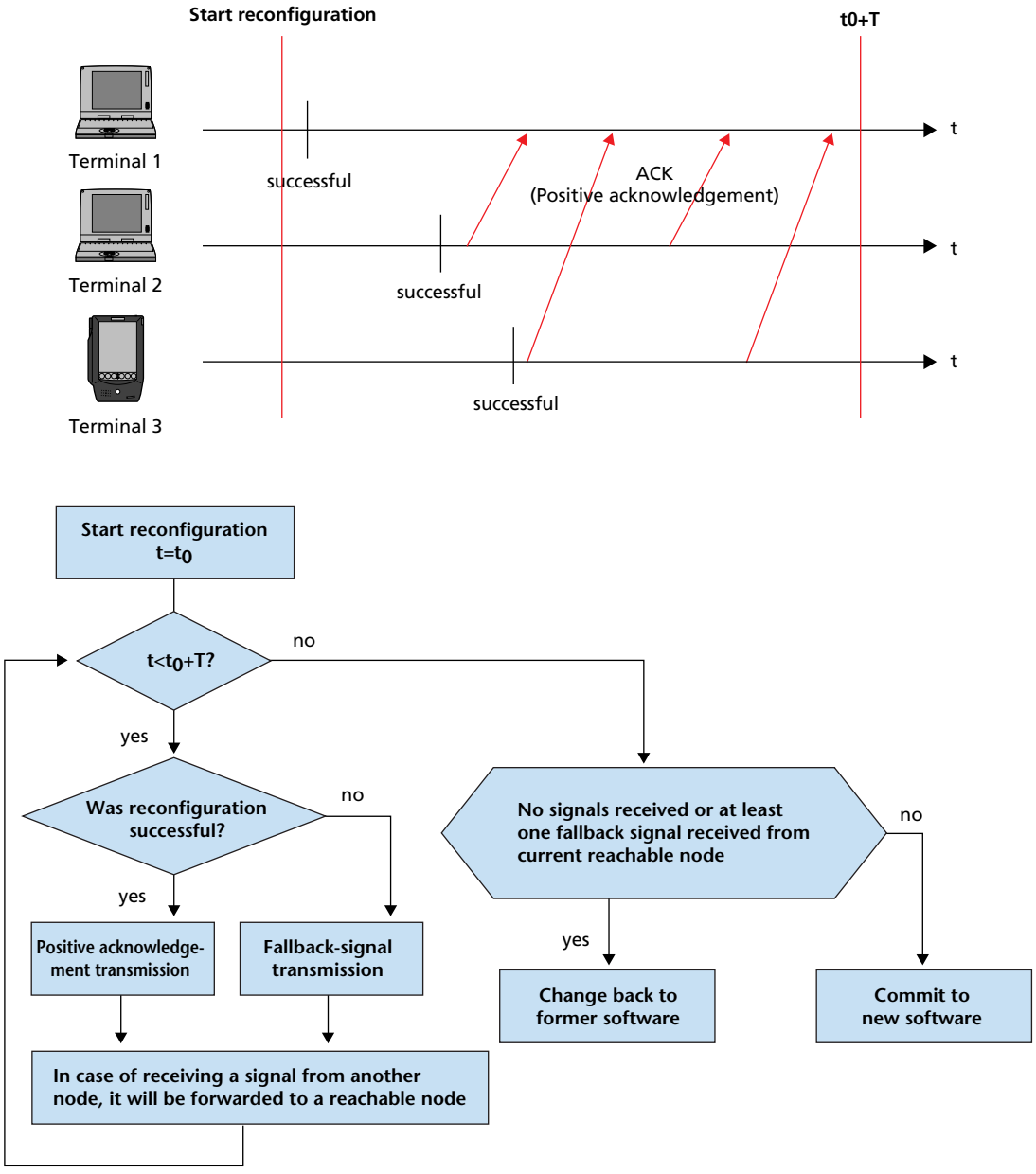

Figure 1 Algorithm and scenario for time-out-based reconfiguration (example: change of routing protocol)

rative research projects as part of our European Framework Programme.

Since April 2002, we have been participating in the Smart user-Centric cOmmUnication environmenT (SCOUT) project on reconfigurability and software radio as part of the Fifth European Framework Programme. Our main contributions to this project are as follows.

- User-friendliness of reconfigurable mobile terminals [3]
- Middleware-centric system architecture for mobile terminals [4]
- Network architecture for reconfigurable mobile terminals [5]
- Reconfigurability in ad hoc networks [1][2]

Since January 2004, we have been involved in the End-to-End Reconfigurability (E2R) project as part of the Sixth European Framework Programme.

3. Service Adaptation

To meet the severe demands placed by services expected to be made available via future mobile networks, it is indispensable to provide a computing environment supporting multiple service platforms. There are several challenges to be faced such as the integration of heterogeneous environments, the shortening of time to market cycles, integration of service components from different providers, runtime software evolution, on-demand personalization, or minimization of system down-time.

Our research on Dynamic Service Adaptation (DSA) [6] investigates technologies to meet the challenges mentioned above.

The goal of DSA is to enable the evolution of multiple services and applications, best described as Unanticipated Software Evolution (USE) [7]. With DSA, services and platforms may evolve in such a way that individual elements of software systems can be selectively modified at a different pace. With that, fields such as service integration, personalization, context awareness, and ubiquitous computing can be advanced. DSA is of importance to future mobile networks on both the network and the terminal side.

Regarding adaptation technology, various approaches, mostly concerned with contents and communication, are being investigated by the research community. In contrast, our research is focused on the adaptation of service logic/behavior. Such adaptation technology can be applied at development time, compile-time, load-time, or runtime. Combining Aspect-Oriented Programming[*1] (AOP) [8], computational reflection [9], and very late binding [10], our research concentrates on the adaptation of services and platforms without services interruption at runtime. The three dimensions of adaptability are shown in **Figure 2**. For example, the requirement to minimize scheduled system downtime can benefit from DSA. Most problems materialize after a system's initial deployment. Up till now, service platforms were unable to address these problems dynamically at runtime. DSA provides mechanisms to deploy adaptation modules into running systems in order to minimize or even avoid their downtime.

The concept of adaptability is closely related to that of modularity [11] and variation points [12]. Modularization is a mechanism for improving the flexibility and comprehensibility of a system while allowing the shortening of its development time. Variation points allow us to explicitly designate module boundaries in a system's design where changes are expected to happen without the need of explicitly naming these changes. Variation points are introduced to support flexibility as a result of the separation and composition of common and variable system parts. Variations and variation points depend on the modularity mechanism offered by the programming platform a system is built on. Most newly built systems are based on object-oriented technologies with classes and instances as modularity constructs as well as units of change. AOP provides a new, more fine-grained, modularity construct that allows us to represent crosscutting concerns, down to the methods of individual

*1 Aspect-Oriented Programming (AOP): A programming technique for improving the separation of concerns in software. It enables clearer expression of programs and program compositions by introducing a new modularity construct called "aspect" that allows to capture crosscutting concerns.

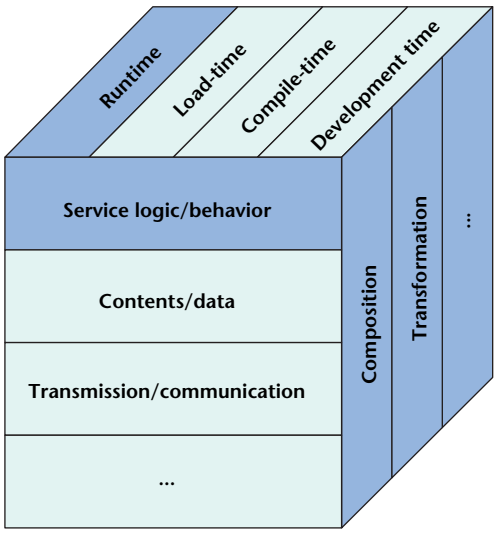

Figure 2 The three dimensions of adaptability

instances.

Our current DSA research platform is based on Smalltalk/Squeak [13] and AspectS [14] (AOP for Smalltalk/Squeak) supporting full computational reflection and very late binding. Parts of our DSA technology proposal have been evaluated via prototypical implementations [6] that allow for debugging, placement and removal of advertisements, the enforcement of style guides, as well as the integration of additional service functionality–all that into a deployed and active running system. A schematic of the adaptation platform utilizing DSA is shown in **Figure 3**.

The next steps in our research on DSA will involve the extension of our platform to cover multiple execution environments as well as the distribution of adaptation modules mentioned in the previous chapter. DSA research covers software engineering principles and mechanisms for software evolution in mobile communication systems. Together with our research partners, we are contributing to and taking advantage of the momentum of the Aspect-Oriented Software Development (AOSD) [15] and USE research communities.

4. Adaptive User Interfaces

In the future, mobile networks will provide access to a wide range of services, such as information access, remote monitoring, and device control from various devices (i.e., mobile terminals or ubiquitous devices); consequently, user interaction with devices and services will dramatically increase.

In that context, future user interfaces should support such a service environment and the following issues should be addressed:

- Various interaction modalities and procedures will be enabled so as to provide more flexibility for the user. Future user interfaces should use context information[*2] to enable adequate simplification and personalization. User's feeling and attention should be one of the context information to consider.
- Supporting user mobility: user profiles should be accessible from various devices. In a ubiquitous environment, users should be aware of the availability of the surrounding devices as well as the functions and services they can support.
- User interaction with different services is difficult when they are activated simultaneously. To enable smooth transitions between services, we should minimize explicit interactions and take advantage of possible implicit interactions.

*2 Context information: Any kind of information that can be used to characterize the state of an entity. An entity may be a person, place, or object relating to the human-computer interaction.

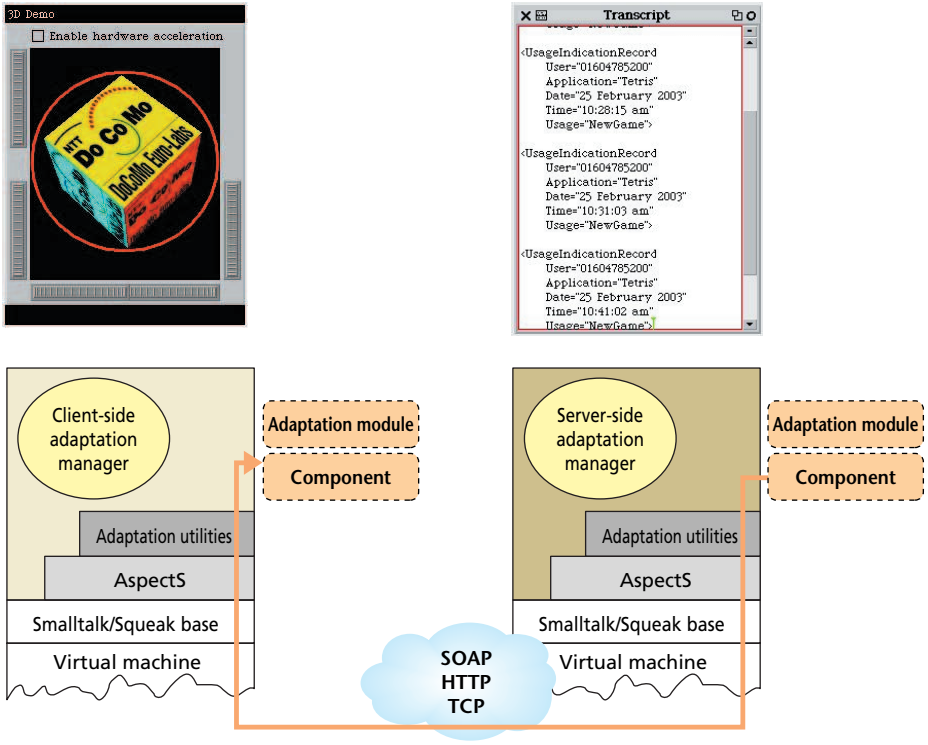

Figure 3 Adaptation platform

To address these issues, we are focusing on user-interaction techniques for controlling simultaneous use of services, and we are investigating so-called notification technologies (description of user scenarios, system requirements, and notification algorithms).

The investigated scenarios [16] [17] are described below. Information about a user in a particular environment (i.e., social, physical, or computational resources)—namely, context information—is gathered from various devices and sensors. The feeling and attention of the user are important aspects of user context. The analysis of context information can be done on the basis of a user-behavior model. Once processed, context information could be part of the user's profile and thus accessible by any device within the system. In this analysis, it is presumed how the user's attention is distributed between different interactions currently engaged in. By foreseeing the attention sharing between future possible interactions, the user interface as well as services and applications could be effectively adapted, and suitable notification means could be used for enabling smooth transitions of the user's attention. Services drawing a user's attention would be augmented using suitable means whereas others may be temporarily suspended.

User feelings concerned with periphery (i.e., things that are noticed without explicitly paying attention to them) are being investigated for notification purpose [17]. Since people are only passively sensing their surrounding atmosphere, it is then necessary to restore their attention when explicit interaction is required. However, peripheral information provides users with an overall understanding of a situation without requiring excessive attention. User's peripheral attention is also considered for realizing a ubiquitous computing environment and an augmented reality environment [18][19]. However, in some cases a user's current focus on a task needs to be interrupted (e.g. by an alarm), so various notification means needs to be considered. To perform notification, a network function such as tracing objects of user's attention is necessary as well. Notification can also mean pre-negotiation in order to perform transition of attention sharing, so the user's confirmation is necessary for the scenario mentioned above [16].

Since January 2004, we have been participating in the Secure, Internet-able, Mobile Platforms LeadIng CItizens Towards simplicitY (SIMPLICITY) project as part of the Sixth European Framework Programme. We are contributing to the definition of user scenarios, technical requirements, and the system architecture concerning a user interface suitable for future mobile networks. In that project, we are also investigating a simply personalized user interface for supporting users with a various range of mobility—which poses another challenge facing a user interface.

5. Conclusion

This article presented the results and current state of our research on sustainable evolutionary systems, namely reconfigurability, service adaptation, and adaptive user interfaces.

REFERENCES

[1] L. Yao and C. Prehofer: "Local Software Update for Terminal Reconfiguration using Ad Hoc Networks," SDR '03 Technical Conference, Nov. 2003.

[2] C. Prehofer and B. Souville: "Synchronized reconfiguration of a group of mobile nodes in ad-hoc networks," ICT '2003, Vol. 1, pp. 400–405, Feb. 2003.

[3] J. Hamard, G. Conaty, and R. Navarro-Prieto: "A User-Centric Approach for the Development of Usable Reconfigurable Terminals," 1st Summit 2003, Vol. 2, pp. 842–846, Jun. 2003.

[4] N. Georganopoulos, T. Farnham, T. Schoeler, R. Burgess, P. Warr, Z.Golubicic, J. Sessler, F. Platbrood, B. Souville, and S. Buljore: "Terminal-Centric View of Software Reconfigurable System Architecture and Enabling Components and Technologies," IEEE Communications Magazine, May 2004.

[5] C. Prehofer, L. Yao, K. Kawamura, and B. Souville: "Middleware and Networking Support for Re-configurable Terminals," SDR '02 Technical Conference, Vol. 2, Nov. 2002.

[6] R. Hirschfeld, K. Kawamura, and H. Berndt: "Dynamic Service Adaptation for Runtime System Extensions." In: WONS '04 Proceedings, LNCS 2928, pp. 225–238, Springer, 2004.

[7] Unanticipated Software Evolution homepage (htlp://www.joint.org/use/)

[8] G. Kiczales, J. Lamping, A. Mendhekar, Ch. Maeda, C. V. Lopes, J.-M. Loingtier, and J. Irwin: "Aspect-Oriented Programming." In: ECOOP '97 Proceedings, 1997, pp. 220–242.

[9] P. Maes: "Concepts and Experiments in Computational Reflection." In: OOPSLA '87 Proceedings, pp. 147–155, 1987.

[10] A. Kay: "Is "Software Engineering" an Oxymoron?" Viewpoints Research Institute, 2002.

[11] D. L. Pamas: "On the Criteria To Be Used in Decomposing Systems into Modules." In: Communications of the ACM, Vol. 15, No. 12, pp. 1053–1058, Dec. 1972.

[12] K. Czarnecki: "Generative Programming: Principles and Techniques of Software Engineering Based on Automated Configuration and Fragment-Based Component Models." Dissertation, TU Ilmenau, 1998.

[13] D. Ingalls, T. Kaehler, J. Maloney and S. Wallace, A Kay: "Back to the Future: The Story of Squeak, a Practical Smalltalk Written in Itself" In: OOPLSA '97 Proceedings, pp. 318–326, 1997.

[14] R. Hirschfeld: "AspectS—Aspect-Oriented Programming with Squeak."

In: M. Aksit, M. Mezini, and R. Unland, editors, Objects, Components, Architectures, Services, and Applications for a Networked World, LNCS 2591, pp. 216–232, Springer, 2003.

[15l Aspect-Oriented Software Development homepage (http://www.aosd.net/)

[16] J. S. Shell, T. Selker, and R. Vertegaal: "Interacting with Groups of Computers." In: Communications of the ACM, Vol. 46, No. 3, pp. 40–46, Mar. 2003.

[17] D. S. McCrickard and C. M. Chewar: "User Goals and Attention Costs." In: Communications of the ACM, Vol. 46, No. 3, pp. 67–72, Mar. 2003.

[18] M. Weiser and J. S. Brown: "The coming age of calm technology. revised version of Designing calm technology." Power Grid Journal, Vol. 1.01, Jul 1996.

[19] G. D. Abowd, E. D. Mynatt, and T. Rodden: "The Human Experience." In: IEEE Pervasive Computing, Vol. 1, No. 1, pp. 48–57, Jan.–Mar. 2002.

ABBREVIATIONS

ACK: ACKnowledgement
AOP: Aspect-Oriented Programming
AOSD: Aspect-Oriented Software Development
DSA: Dynamic Service Adaptation
HTTP: HyperText Transfer Protocol
SCOUT: Smart user-Centric cOmmUnication environmenT
SIMPLICITY: Secure, Internet-able, Mobile Platforms LeadIng CItizens Towards simplicitY
SOAP: Simple Object Access Protocol
TCP: Transmission Control Protocol
USE: Unanticipated Software Evolution