Interfering Effects of Adaptation: Implications on Self-adapting Systems Architecture

Jacqueline Floch, Erlend Stav, and Svein Hallsteinsen

SINTEF ICT NO-7465 Trondheim, Norway
{jacqueline.floch, erlend.stav, svein.hallsteinsen}@sintef.no

Abstract. When people are moving around using handheld networked devices, the environment for the provided services vary influencing service quality properties and user needs. In order to maintain usability and usefulness for mobile users, dynamic service adaptation is needed. Several forms of adaptation may be applied. For example, the application structure may adapt from thin client to self-reliant client, or network handover may be performed. The selection of an adaptation type is however far from obvious. Adaptation usually has impact on system resources or service quality. Also, one adaptation may require other adaptations that again have impact on resources and quality. This paper illustrates the complexity of selecting an adequate adaptation form. We argue that adaptation selection requires advanced reasoning and identify implications on the architecture of self-adapting systems.

1 Introduction

When people are moving around using handheld networked devices, the operating environment for the provided services vary influencing service quality properties and user needs. To retain usability, usefulness, and reliability under such circumstances, systems should adapt to the changing environments. Service adaptation is about finding the application configuration that best fits the context, where context includes both system context such as battery level and network resources, and user context such as position, noise and user needs. Adaptation may be performed at several levels and in different ways. Adaptation may be applied on the applications, or on the resources and devices required by the applications. It may require modifications to the application structure, to the selection of application components, or to their deployment. A close analysis of the problem of adaptation shows that the selection of the "best configuration" is complex and requires reasoning on dependent context elements and adaptation forms. This paper presents a mobile service scenario that illustrates this complexity and draws out a set of requirements on the architecture of self-adapting systems. The MADAM project is currently developing solutions based on the requirements derived from the scenario analysis [1]. Current approaches to self-adaptation usually describe abstract motivations. The research literature lacks presentations of scenarios that would provide a common base for understanding business problems, extracting valuable business requirements, and justifying the research problem relevance.

2 Scenario Example

The application domain for our scenario is inspection and maintenance support for janitors. Janitors use handheld networked devices during their work. They are involved in various working situations, ranging from administrative work in a quiet and connected office environment, through travelling between technical installations in rugged industrial environments with varying network coverage. We assume that the companies where inspection is performed make an intensive use of ICT systems for registering information, tagging and controlling equipment. In the following, our scenario is structured in a set of scenes that relate to various working situations and contexts.

Scene 1 – Morning at home: The janitor checks his assignments for the day before he leaves home, using his company planning application on his handheld device. He is also running a video player on the device showing morning news on a screen in the kitchen. The first assignment is about fixing a ventilation system in a large building. He starts looking at information about the first assignment. There is little memory because the video player uses a lot. The home WLAN provides high capacity network connection. The device has been charging during the night and is still connected to outlet power, so power is abundant and the load on the server is low. In this situation, a thin client configuration is chosen as the initial configuration of the work planning application.
Scene 2 – Leaving home: The janitor shuts down the video player and prepares to leave home. This new situation raises a relevant context change: the memory available on the handheld becomes high. In order to increase application response time and reliability, the work planning application is reconfigured with a richer client caching data to save power on the handheld and to become less vulnerable to network instability.
Scene 3 – Driving: The janitor enters his car to go to the company building where the faulty installation is located. While he is driving the janitor wants to check more details, but since his eyes and hands are busy with the driving, he prefers hands-free user interface. The janitor selects tools and information for guiding the inspection assignment and initiates their downloading. The device is now connected through GPRS to the janitor company server. As the cost of using GPRS is high and the capacity of the network low, the downloading of the tools is postponed.
Scene 4 – Arriving at the customer: When the janitor arrives to the company site, he gets access to the company WLAN. He can now download tools. However the network cannot be regarded as trusted, and a VPN tunnel has to be established.
Scene 5 – Measurement: The janitor starts the inspection of the ventilation system. He starts the inspection application and is guided around in the building to measure temperature. During the work, he has to deal with different kinds of temperature sensors. Measurement is performed manually or automatically using Bluetooth. In the later case, various sensor drivers are needed depending on the sensor types. Drivers can be downloaded from the company equipment server. The building under inspection is large and the measurement collection has already lasted a long time. In order to reduce battery consumption, the measurement application switches to a stand-alone mode and

Table 1. Scenario: adaptation summary in a situation-action style

Scene	Relevant context and context changes	Adaptation
1	**Handheld:** available memory: low; high battery level **Network:** WLAN: high bandwidth, low cost	The initial configuration is selected. A thin client configuration is chosen.
2	**Handheld:** available memory: high	The application is reconfigured from thin to caching client. Assignment information is downloaded.
3a	**User needs:** hands-free mode	A hands-free UI is added.
3b	**User:** location: driving to customer **Network:** GPRS: medium bandwidth, high cost	The downloading of inspection tools is postponed
4	**User needs:** security policy **Network:** WLAN: high bandwidth, low cost	A VPN tunnel is established. The downloading that was postponed is started.
5a	**Infrastructure:** new sensor	The sensor drivers are downloaded and installed.
5b	**Handheld:** rapidly decreasing battery level **Network:** WLAN: high bandwidth, low cost	The application is reconfigured from a network connected mode to stand-alone mode. The data measurements are saved periodically
6	**User:** application priority **Handheld:** available memory: low **Network:** WLAN: high bandwidth, low cost	The data measurements are saved and the inspection application is suspended. The planning application is started.
7	**Infrastructure:** new computer	The inspection application is redeployed.

measurements are stored locally. However, the network coverage is good and measurements data are saved centrally periodically.
Scene 6 – Notification: During the measurement activity, the janitor is interrupted by a notification about a new task. The planning support application requires more resources than currently available on the handheld. In order to enable planning, measurements data are saved to the company equipment server, and the measurement application is partially suspended.
Scene 7 – Measurement analysis: When all measurements are collected, the janitor moves to the technical office where he can use a more powerful stationary computer to perform measurement analysis. When he enters the office, the janitor work session is automatically moved from the handheld to the stationary computer.

Table 1 summarizes the scenario in a situation-action style where each situation leads to an adaptation action. In that simple scenario, we observe that each situation requires taking into account various kinds of context. We also observe that various adaptation forms such as adaptation of functional richness, adaptation of behaviour and data deployment, and adaptation of the user interface modality, may take place.

3 Adaptation Effects

While Table 1 describes simple relations between situations and adaptation mechanisms, this section provides a deeper analysis demonstrating the complex dependencies between adaptation and context, and the effects of adaptation on system resources and offered service quality. We do not restrict to the single scenario, but generalize adding new context conditions that may occur under the janitor work.

Table 2 presents the analysis in a goal-oriented style. A goal describes a high-level behaviour objective that the self-adapting system should attempt to fulfil in order to maintain service usefulness and quality when context changes occur. Usually several adaptation mechanisms may be applied to achieve a goal. A classification according to goals allows us to present the relations between context and adaptation mechanisms in a concise way. Table 2 distinguishes between "primary context elements" i.e. the main triggers for adaptation, and "secondary context elements" that complement the primary elements when making a decision about adaptation. The "adaptation effects" describe the impact of adaptation: "(C)" indicates an impact on context, "(S)" on service quality, "(G)" on other goals, and "(A)" indicates an inferred new adaptation need.

Table 2. Adaptation analysis in a goal-oriented style

Goal	**Context**	**Adaptation mechanism**	**Adaptation effect (s)**
Maintain service availability	**Primary:** low power level **Secondary:** availability of external, handheld device or PC	Redeploy application session	(A) Adapt application configuration to new platform
	Primary: network coverage/no coverage	Redeploy application and data	(S) data integrity
Enhance operability	**Primary:** user activity, hands occupation **Secondary:** audio capabilities	Select UI modality (e.g. voice or text based UI)	(C) handheld resources consumption
	Primary: equipment, device and service extensions **Secondary:** network coverage (e.g. Bluetooth)	Enrich application functionality	(C) handheld resources consumption
		Launch new application automatically depending on extension type	(C) handheld and network resources consumption
Control power consumption	**Primary | Secondary** user activity duration **Secondary | Primary** limited power resources	Adjust power demanding operations: network access	(A) Redeploy application; tune data synch. (C) network resources consumption
		Adjust power demanding operations: CPU frequency	(S) service response time

Table 2. (*continued*)

Goal	**Context**	**Adaptation mechanism**	**Adaptation effect (s)**
Optimize memory usage or Optimize CPU usage	**Primary** memory/CPU resources	Redeploy application (client / server split)	(S) service response time; data integrity (C) network resources consumption
		Select media type and richness adapted to resource	(S) service accuracy
	Primary memory/CPU resources **Secondary** priority of user tasks	Suspend low-priority applications	(G) service availability
Select a satisfactory network	**Primary** available networks (e.g. GSM, WiFi) **Secondary** user/application needs (e.g. cost, response time, security)	Hand over (switch) between networks	(C) resource consumption (S) cost and provided QoS
		Select a network adaptor adapted to network	(C) resource consumption
Optimize network usage	**Primary** network capacity	Redeploy application (client / server split)	(S) service response time, data integrity (C) power consumption
		Select appropriate time to perform operations (e.g. postpone task)	(G) service availability

		Adjust data richness;
		select media type;
		tune data synch.
			(S) service accuracy

	Primary network security	Select the appropriate security model (e.g. VPN, encryption level)	(C) resource consumption (S) response time

4 Implications on System Architecture

By illustrating the interfering effects of service adaptation, the analysis presented in table 2 demonstrates the complexity of developing adaptive applications. In this section, we extract a set of implications on the architecture of self-adapting systems. These implications relate to the main functionality necessary for adapting applications: context monitoring, adaptation reasoning and reconfiguration.

Firstly, we observe the complexity related to context monitoring. Multiple context elements need to be taken into account. Further, these span from elementary elements, such as network cost, to more complex aggregated or derived elements, such as predicted location. Most of these elements are domain independent. We expect the set of relevant elements and the sources producing them to evolve in the same way as applications. This gives the following architectural implications: i1) *Context monitoring should be kept separate from the application and realized through reusable components* *or context middleware*. i2) *The context middleware should be extensible and support the addition of new elements and new forms of reasoning*.

Secondly, concerning adaptation reasoning, we observe multiple relations between context and adaptation mechanisms, and interfering effects of adaptation. During the generalization done in section 3, we found it difficult to capture all relations. We also expect that new relations will be introduced as applications and context monitoring evolve. Two main approaches [2] have been proposed for self-adaptation: internal approaches where adaptation is realized as part of the application using programming language features, and external approaches where adaptation mechanisms are realized by an application-independent middleware. The main drawback of internal approaches is the complexity introduced by intertwining adaptation and application behaviours. Also, they poorly support application and adaptation evolution. Given our observations, these drawbacks make internal approaches inappropriate in the context of mobile services, and thus: i3) *Adaptation mechanisms should be realized externally to the application*. External approaches require adaptations policies to be described separately from the applications. These policies are used by the middleware to reason and decide about adaptation. Three main approaches have been proposed for the description of policies. Two of them are respectively illustrated by table 1 and table 2: situation-action approaches [3] and goal-oriented approaches [4]. The third approach uses utility function that assign a utility value to each application variant as a function of application properties, context and goals [4]. The interfering effects of adaptation make the two first approaches inappropriate, and thus: i4) *Adaptation policies should be expressed using utility functions*.

Finally, concerning reconfiguration, we need to build adaptable applications. Two general approaches have been proposed [5]: parameterization supports fine tuning of applications through the modification of program variables, while compositional variability is specified at the component level allowing the modification of application structure and algorithms. Parameterization is an effective way to implement variability, but may also lead to a large set of variants and raise scalability issues, implying: i5) *Adaptable applications should be built on compositional variability combined with cautious use of parameterization*.

A main challenge given these implications is to develop effective and scalable solutions for handheld devices with restricted processing and memory capabilities.

References

1. MADAM "http://www.ist-madam.org/"
2. Oreizy, P. et al. "Architecture-based approach to self-adaptive software", IEEE Intelligent Systems and Their Applications, 1999, vol. 14 (3).
3. Garlan, D. et al."Rainbow: Architecture-based self-adaptation with reusable infrastructure", IEEE Computer, 2004, vol. 37 (10).
4. Kephart, J.O. and Chess, D.M. "The vision of autonomic computing", IEEE Computer, 2003, vol. 36 (1).
5. McKinley, P.K. et al."Composing adaptive software", IEEE Computer, 2004, vol. 37 (7).