Quality of Adaptation

Eli Gjørvø, Frank Eliassen
Simula Research Laboratory
P.O. Box 134, N-1325 Lysaker, Norway
{eligjfran}@simula.no

Abstract

In order to support reusable and extendable solutions to self-adaptation, state of the art within adaptive systems include a diversity of architectures and mechanisms targeting different application types, technologies, context and concerns. Current approaches do not allow reasoning about the mechanisms and strategies themselves during start-up time or run-time. Consequently, the configuration or tuning of the adaptation system itself, must be done manually by a user or an administrator, or hard coded at design-time, making it harder to reuse.

In this position paper, we propose considering adaptation as a service. In order to allow reasoning about and comparing such services, we define Quality of Adaptation (QoA) to be the qualitative properties of an adaptation service. We argue that this approach facilitates the development of adaptation mechanisms and control that are, like the system they adapt, reusable and adaptable, and open to evolution.

1 Introduction

Systems are increasingly expected to adapt themselves to changing requirements and environments providing dependability, robustness and availability with minimum human interaction. According to [16], a self-adaptive system is one that i) supports the continuous process of monitoring and configuring the system in order to adapt at run-time to the changing operational environment and ii) supports system evolution, which is the consistent software change over time. State of the art within adaptive systems includes a diversity of architectures and mechanisms targeting different application types, technologies, context and concerns. However, the developer of the adaptive system must make many difficult, but important, decisions at design time. These decisions determine the run-time properties provided by the resulting adaptation system, for example its performance, how it preserves the correct behavior of the application and if it considers security issues. Furthermore, they influence the possibilities to allow the adaptive system to evolve, that is to be extended and modified as new needs and requirements are introduced at later points in time.

First, the developer must make some assumptions about the application that the system is going to adapt, and its requirements. Thus, the adaptation system will be difficult to reuse when these assumptions do not hold, and hard to extend when new requirements are introduced. As an example, [17] presents a dynamic configuration algorithm based on searching through alternative configurations and the corresponding resource needs using an approximation algorithm. The experiments show that the algorithm performs well when applied to the selection of the combination of a few applications (a text editor, a video player and a browser) and the number of alternative configurations is low (30). However, in the case of configuring finer grained services, such as advanced video streaming applications that require parameter tuning of video processing, encoding and lower level communication components [5][13], the number of alternatives could be several thousands. Searching through the configuration space might not be feasible, even if using an approximation algorithm.

Second, as adaptation has interfering effects on the adapted system, the functional and qualitative requirements to the adapted systems impose corresponding functional and qualitative requirements to the adaptation system [7]. For example, applying a search algorithm such as the above-mentioned on a large set of alternative configurations may consume a substantial amount of computing resources like CPU and memory. The extensive resource consumption may cause the application itself to slow down, or even be swapped to disk, while the adaptation algorithm is running. In order to satisfy the application requirements, the adaptation mechanisms and strategies must be configured and tuned towards the specific application and the run-time context of the system.

Third, during run-time, events may occur that causes the application requirements to change, for example applica-
tion mobility, where an application is migrated from a laptop to a resource constrained PDA, device mobility, where a device moves from a secure area to an insecure, introducing the need for security management, or new user needs or preferences, where the user decides that he wants more control over the adaptation. Consequently, there is a risk that the current adaptation behavior may not match the new requirements, and that some or all of the adaptation strategies and mechanisms initially selected must be replaced.

Current approaches do not allow reasoning about and configuring the adaptation system itself, neither at system initialization nor at run-time. Thus, configuration of the adaptation system must be done manually by a user or an administrator, or it must be hard coded into the adaptive system, either way making it harder to reuse. We argue that the adaptation mechanisms and strategies themselves, like the applications they adapt, should be reusable and evolvable in the sense that they should be applicable to a large and extendable set of applications, contexts and users.

1.1 Adaptation Terminology and Scope

Terms like “adaptation”, “adaptation mechanism”, “adaptation strategy” do not have consistent definitions within the research community. In this paper, we include in adaptation both initial system configuration and system reconfiguration, and we consider adaptation as a process where the following steps are performed.

Adaptation must be triggered, that is some entity in the system must initiate the adaptation process. In many current approaches, adaptation is triggered based on context information that is compared with user and/or system requirements. Adaptation planning is the process where alternative adaptation actions are considered and compared. Examples of adaptation action are to replace component A with component B, or migrate component A to a new location. When the alternative actions have been analyzed, decision making is applied to decide which actions will be performed, if any. A decision making algorithm in effect implements an adaptation strategy. Context information may also be used during adaptation planning and decision making. Configuration is the process of executing the adaptation actions using adaptation mechanisms such as loading components from local or remote repositories, destroying components that are no longer used, creating and destroying bindings, and transferring state between components.

Under this definition, adaptation has a broad scope. It is not specified when or why adaptation should take place, or how it should be performed. Furthermore, we include both adaptation that affect one application independently, and adaptation that affect a set of applications. An example of the former, is the replacement of one of the components in an application. An example of the latter, is to insert a security policy that forbid making insecure connections to certain port numbers.

1.2 Adaptation as a service

We believe that by considering adaptation a service, we may reuse principles and mechanisms developed for other types of services in order to achieve reusability and adaptability also for adaptation services. We define the term Quality of Adaptation in order to allow run-time reasoning about the adaptation services and their effects, which is necessary in order to be able to configure or tune the adaptive services towards the running applications and their context.

This paper is organized as follows. Section 2 contains an overview of research work that discusses issues related to the design and implementation of adaptive software as described above, and Section 3 describes in more detail our approach. Section 4 contains an analysis of the ideas presented in Section 3, and finally, in Section 5, we present summary and further work.

2 Related Work

When studying state of the art within self-adaptive systems, we observe that many of the current approaches address some of the issues mentioned in Section 1.

2.1 Adaptation as Reusable Components

In the QuO project [23], a middleware solution to supporting end-to-end QoS management in distributed real-time embedded systems has been developed. Specialized QoS management functions are encapsulated by so-called Qoskets.

In [20], it is described how individual off-the-shelf Qosket Components can be configured together with the application at assembly time by the QuO middleware. Qosket components are component implementations of Qoskets using component models such as the CORBA Component Model, and that provide a set of interfaces that can be used by the middleware layer in order to control them.

Qosket components provide a general solution to manage end-to-end QoS requirements without requiring the middleware to provide such services. However, in order to be able to fully take advantage of such components, there is a need for a framework that is able to reason about the needs for QoS management functions at run-time, and to map these needs to concrete qosket components, and patterns for how to use them.
2.2 Qualitative Properties of Adaptation Systems (Classification)

As part of the discussion of their approach, several research approaches address some qualitative properties of the adaptation system itself. In Table 2 we give an overview of some of these projects, and some qualitative properties addressed by each project. A D in the table means that the property is discussed, while an S in the table means that the referred papers also describe solutions to achieving the property. Included in the table are also two survey papers that includes discussion about some of the properties. The properties mentioned in the table has the following meaning:

We base our use of the term “safe adaptation” on how it is used in the area of self-adaptive systems: an adaptation that does not cause damage to the system it adapts. For example, in [21], “safe adaptation” means that the program maintains its integrity during adaptation.

We use “secure adaptation” to mean an adaptation that does not contribute to decreasing the adapted system’s resistance towards attacks, for example by introducing insecure links to a secure application.

By “optimal adaptation” we mean an adaptation that finds optimal solutions to adaptation problems. For example, an exhaustive search algorithm like the one mentioned in Section 1, eventually finds the optimal solution in all cases, while an approximation algorithm in some cases returns a suboptimal solution.

By “the cost of adaptation” we mean the amount of resources like CPU cycles, memory, or money, consumed by adaptation relative to the total amount. For example, an exhaustive search algorithm potential that consumes a lot of CPU and memory, imposes a high cost on a resource constrained device. This use of the term is based on the observation that in the theoretical case where we have infinite computing resources, any resource consumption is not a cost.

By “the performance of adaptation”, we mean the performance of the adaptation system itself, for example how many seconds it takes.

By “human in the loop” we mean what kind of human intervention is required or allowed by the adaptation system. For example, the system could require no human intervention, as is the vision of the autonomic computing community [11], it could only inform the user but allow no influence, or it could require feedback from or provide control to the user in certain situations.

From Table 2, we see that the properties described above are addressed by many of the projects. However, they are addressed independently, despite the fact that the properties may affect each other at run-time. There are no current solutions that we are aware of that discusses qualitative properties of adaptation from a general point of view. In the context of self-managing systems, we believe a holistic view of all dimensions is necessary in order to reach system and user goals.

3 Adaptation as a Service

In this section, we elaborate on the idea of regarding adaptation as a service itself, and we discuss the importance of the qualitative properties of adaptation services.

3.1 Adaptation Service Definition

In line with service oriented computing [12], we characterize an application by the services it provides to its users. We define a service as the “work done”, specified
by a subset of output events, and the causally related “work request”, specified by a set of inputs, to some composition of components [19]. A service may be as simple as to calculate the sum of two numbers, or more complex, such as delivering video and audio content to users during a video conference. Furthermore, a service can be composed of subservices in a recursive structure, and it may explicitly define dependencies to other services.

![Figure 1. Adaptation service.](image1)

If we apply our definition of service to the adaptation process described in Section 1, an adaptation service can be defined as a service whose input event is an adaptation trigger, and whose output events are a set of services that potentially has been modified or produced during the adaptation. This definition is illustrated in Figure 1. The definition does not restrict the goal of the adaptation process. In many situations the goal of adaptation is to maximize the user satisfaction. However, there could be other reasons for performing adaptation. For example, in order to maximize profit, a service provider could perform adaptation with the goal of increasing the number of paying customers, even though this means that each customer gets a poorer service.

![Figure 2. An adaptation service composed from a set of subservices.](image2)

As we allow services to be composed of subservices, so can adaptation services. The adaptation process described in Section 1.1 can be decomposed as shown in Figure 2 into the subservices Adaptation Planning, Decision Making and Configuration, that again consists of one (or more) Adaptation mechanisms. Furthermore, the subservices Adaptation Planning and Decision Making may depend on Context sensing and reasoning services as also shown in the figure.

By considering adaptation as a service, we implicitly turn the adaptation services themselves into potential subjects for run-time adaptation. Consequently, the adaptation process described above could be reused to reason and adapt the adaptation services themselves.

### 3.2 Quality of Adaptation (QoA)

In order to define what Quality of Adaptation (QoA) is, we first discuss our definition of the commonly used term Quality of Service (QoS). We base our understanding of QoS on the definition used in the QuA project, as a function of the distance from the (theoretical) ideal service output [18][19]: Given a trace of input events to a service, the ideal output trace is generated when the service executes completely and correctly on an infinitely fast platform with unlimited resources. The distance from this ideal is measured using a set of dimensions by which the qualitative behavior of the service can be described. For example, in order to measure the quality of a component encoding raw video frames, we may consider the framerate delivered by the encoder relative, and the resolution of the output frames. The details of this model can be found in [19].

Using the definition of adaptation defined in Section 3.1 above, QoA is used as a measure for how the output from an actual adaptation deviates from the output of a corresponding ideal adaptation service. According to the definition of ideal output above, the ideal output from an adaptation service is generated when the adaptation process executes completely and correctly on an infinitely fast platform with unlimited resources. The dimensions used to measure QoA can be related to the resulting output services, for example that they execute correctly both during and after the adaptation has been performed, or to the adaptation process itself, for example the memory and CPU cycles used by the adaptation process.

### 3.3 Example

In order to illustrate the ideas presented above, let us consider the example of an adaptable video conference between two participants, each using a portable device connected to a wireless LAN (WLAN). Since the discussion is private, the participants do not want their conversation to be heard by others. We assume an adaptation service similar to the one depicted in Figure 2. Adaptation mechanisms are used to insert or replace communication components and video/audio encoding/decoding components, and to set parameter values in these components.

In order to describe the quality of adaptation services for this application, we can measure the adaptation ser-
vice along the QoA dimensions described in section 2. The safety of adaptation can be measured as the deviation from seamless configuration of the audio and video stream, in terms of the delay or jitter introduced during configuration. The security of adaptation can be measured as the increase, introduced by an adaptation process, in the probability of an attacker succeeding in eavesdropping on the application. The optimality of adaptation can be measured as the probability that the planning algorithm selects non-optimal configurations and configuration parameters. The performance of adaptation can be measured as the number of seconds it takes to perform an adaptation, and the cost as the percentage of the total amount of memory used by the adaptation algorithm. Finally, human in the loop, can be measured as how often the user’s preferences related to his influence on adaptation, are violated.

These dimensions allow to measure the adaptation system’s deviation from the ideal, and the dependencies between the QoA dimensions. For example, in reality, adding encryption impose a cost in computation. It may be necessary to trade QoA dimensions against each other. For example, if privacy is important, cost can be traded for security. Furthermore, the priority of QoA dimensions may change at run-time. For example, if the application is moved from a laptop to a resource constrained PDA, it may be necessary to switch from an adaptation strategy that always finds optimal solutions to one that prioritizes the computational cost of the adaptation algorithms.

4 Evaluation and Analysis

4.1 Benefits of Adaptation Services

Developing adaptation as a service adheres to the widely accepted principle that software should be developed as modules that explicitly define their behavior, and encapsulate their implementation details. This contributes to making the adaptive behavior more reusable, since it separates the implementation of the adaptive behavior from the implementation of the application. Even though complete separation is not possible, the ability to describe both functional and qualitative properties enables reasoning about the quality of a certain adaptation service for a specific application in a specific run-time context. It also enables reasoning about the different behaviors of updated versions of the adaptive software, and their distinctive features, thereby contributing to supporting evolution.

As pointed out in Section 1, adaptive behavior may have interfering effects on the adapted system. Developing adaptation as a service does not eliminate this potential problem, but assuming the possibility to reason about the adaptive services at run-time, it is possible to configure the adaptive services in order to minimize the effect on applications.

It may be difficult to reason about what “good QoA” is. The definition of QoA as presented in Section 3, assumes the quantification or measure of QoA using QoA dimensions. As the example in Section 3.3 demonstrates, it may be necessary to prioritize QoA dimensions, or even trade them off against each other, much similar to QoS trade-offs that are performed in QoS-aware systems. Which dimensions that are most important in order to achieve “good QoA”, varies from system to system, and maybe even from user to user. However, as discussed in Section 2, individual properties of adaptive systems are quite often addressed. Thus, we think the idea of providing a systematic approach to understanding such properties, and the dependencies between them, is intriguing.

4.2 Implementing and Testing the Idea

As middleware solutions such as [1][8][20] facilitate the separation between application concerns and adaptation concerns, we believe that the middleware layer is the most promising place to deploy adaptation services. However, the successful integration of adaptation services into middleware, puts some requirements on the middleware.

In order to take advantage of the adaptation services as they are developed, the middleware can not be constructed as one, monolithic module. Rather, it must be able to plug in adaptation services as they are developed and made available. Furthermore, in order to allow reasoning and tuning the adaptation services themselves, it must also provide a general planning framework that can be applied to any kind of service, not only application services.

We believe that a component model based on computational reflection [1] is well suited to support adaptation services, because reflection enables inspecting and manipulating components at run-time. However, the system must support reflecting on both the functional properties, in terms of functional adaptation behavior, and qualitative properties, in terms of QoA dimensions, of adaptive services.

As the QuA middleware supports these properties [10], we intend to perform experiments using the QuA prototype, in order to validate the ideas presented in this paper. QuA is designed with a minimal core that can be extended with functionality, such as adaptation and reconfiguration, depending on the application needs and run-time context. It uses a mirror-based reflective model that supports reflection on functional and qualitative properties of any service both before and during its execution. Finally, QuA provides basic support for planning services that can be used to plan both the adaptation of applications, and the adaptive services themselves.
5 Summary and Conclusions

In this paper we have proposed a principled approach to designing and developing adaptive behavior based on defining adaptation as a service. We have also defined the term Quality of Adaptation, in order to enable reasoning about both functional and qualitative properties of adaptation services. We argue that this approach facilitates developing adaptation services that are, like the applications they adapt, reusable, adaptable and support evolution.

We have implemented a planning-based middleware, QuA, where middleware services, like decision making, adaptation and monitoring, are developed and plugged into the middleware as services, as explained above. Based on this middleware, we plan to investigate adaptation services, and to explore the possibilities of dynamically reasoning about and modifying such services.

Acknowledgments

We would like to thank KetiL Lund for his significant contribution in developing the ideas and concepts presented in this paper.

References