Managing Distributed Adaptation of Mobile Applications

Mourad Alia[1], Svein Hallsteinsen[2], Nearchos Paspallis[3], and Frank Eliassen[4]

[1] Simula Research Lab, Martin Linges v 17, Fornebu, P.O.Box 134, 1325 Lysaker, Norway
mouradal@simula.no
[2] SINTEF ICT, S.P. Andersens vei 15 b, Trondheim, Norway
svein.hallsteinsen@sintef.no
[3] Department of Computer Science, University of Cyprus, P.O. Box 20537, Nicosia, Cyprus
nearchos@cs.ucy.ac.cy
[4] Department of Informatics, University of Oslo, P.O.Box 1080 Blindern, Oslo, Norway
frank@ifi.uio.no

Abstract. Mobile computing is characterised by variations in user needs and in the computing and communication resources. We have developed a middleware centric approach for the development of software capable of dynamically adapting to such variations. The middleware leverages models of needs and resources and the adaptation capabilities of the software and performs context monitoring, adaptation planning and dynamic reconfiguration at runtime. In this paper we focus on the modelling of resources of a distributed mobile computing infrastructure and how the resource model is used in adaptation planning. We present a distributed resource management framework and mechanisms necessary to maintain an up to date resource model at runtime. The challenge is to balance the level of abstraction so as to hide some of the heterogeneity of the actual infrastructure while retaining sufficient detail to serve the needs of distributed and centralized adaptation planning. The proposed framework is illustrated through a running example.

1 Introduction

With the increasing mobility and pervasiveness of computing and communication technology, software systems are commonly accessed through handheld, networked devices, carried by people moving around. This introduces dynamic variation both in the user needs and in the operating environment of the provided services. For example, communication bandwidth changes dynamically in wireless communication networks and power is a scarce resource on battery-powered devices when outlet power is not available. Under such circumstances, applications need to adapt dynamically in order to retain usability, usefulness, and reliability. To design such applications many recent works have proposed general solutions based on an adaptation loop control monitoring user needs and available resources and adapting the application accordingly. However, most of these solutions concentrate on the dynamic reconfiguration of the mobile application on the client device, without properly exploiting the computing resources available throughout the wireless networks it is mostly connected to [1,2].

In the MADAM[1] approach, adaptations are carried out by generic middleware where earlier implementation of the running application are reconsidered in response to context changes [4]. This is also referred to as planning. The (re)planning is based on runtime architecture models of applications with their adaptation capabilities explicitly modelled. During re-planning, alternative architecture models of the running applications are dynamically generated by considering alternative implementation choices for the component types of the applications. The adaptation reasoning relies on the use of utility functions allowing the computation of the utility for the user of an application variant given the current user needs and available computing and communication resources. The utility functions are designed and implemented with the aim of dynamically measuring the benefit of a given variant.

In this paper, the current MADAM approach is extended to enable the adaptation planning process to leverage the deployment (and possible re-deployment) of the application components in a distributed mobile computing infrastructure. Such an infrastructure will usually consist of several nodes connected via one or more communication networks. Distributed adaptation planning can then be used to increase the performance of the user application and to minimize the latency and the communication overhead – as in classical grid-like infrastructures – or yet to increase the availability of the end user service.

The main contribution of this paper is the combination of a resource management framework and utility-based adaptation reasoning used to manage distributed mobile adaptation. The aims of the resource management framework are (i) the modelling of the resources of the distributed infrastructure so as to hide the heterogeneity of the infrastructure and to provide a uniform and generic method to access the resources and (ii) resource control and management, both locally in one node and globally by maintaining a global view of the distributed computing infrastructure composed of a set of discoverable resources. The adaptation reasoning uses the resource management framework to discover the available and surrounding resources (the different networks, servers, etc.), select the best placement using the utility functions and finally allocate the selected resources with the associated amounts.

We start the presentation of this paper by a motivating example in section 2. Then section 3 presents the overall MADAM middleware-centric approach for the management of distributed adaptation. The resource management framework , as part of the proposed middleware, is presented by the section 4. Section 5 explains how the adaptation manager interacts with the resource manager component to perform distributed adaptation reasoning. An implementation status is given in section 6. Finally, section 7 discusses some relevant related works before concluding in section 8.

2 Motivating Example: On-Site Worker Application

The following scenario shows how self-adaptation is crucial to retain the usefulness of the application across typical context changes. The scenario describes an application, which is used to assist a maintenance worker with on-site work. Because of the nature of her work, the worker may not always be able to visually interact with the mobile

[1] MADAM is a European IST Project [3].

device (e.g. a PDA running the application). Consequently, the application is designed to offer two modes of user interaction: visual and audio interfaces. The visual interaction communicates with the user with visual messages (e.g. text in popup windows) while the audio interaction uses the PDA's speakers and microphone to communicate information to and from the user.

Consider the case where the worker is in her office, interacting with the PDA in order to prepare the application for the on-site visit. In this case, the application has the full attention of the user, and consequently the visual interaction is selected as it is more responsive (e.g. faster interaction) and more resource efficient (i.e. requires less memory and CPU, and no networking). But when the worker starts working on the gear to be maintained, the application switches to audio interaction, thus releasing the worker from the burden of having to visually interact with the application. Finally, when the network is sufficiently fast and cheap, and the resources are low (e.g. because the PDA starts other applications as well), the application switches to the audio mode where the speech-to-text component is hosted by another computer, for example a pc at the site. As illustrated by this scenario, the main effort of the self-adaptation mechanism is to monitor the context (i.e. the status of the hosting device and the user), and dynamically respond to changes by selecting the most appropriate application variant. To keep things simple, this scenario considers three modes only: the visual, the audio, and the audio with remote text-to-speech processing modes.

This scenario demonstrates that certain applications can improve their provided utility by switching between alternative application behaviours and deployment. It therefore stresses the need for a distributed resource management framework for the management of the distributed adaptation.

3 Distribution Adaptation Management

We focus on the resources directly surrounding one mobile user such as the handheld device, the home PC, or the laptop in the suitcase. This collection of computers together with the communication networks they are connected to, is referred to as an adaptation domain Within an adaptation domain all computers run an instance of the MADAM middleware and there is one client (the handheld device) and zero or more servers. The domain is formed dynamically by the means of a discovery protocol, whereby servers regularly advertise their presence, and the client keeps track of available servers. Servers may be shared, meaning that they are members of more than one domain. All the servers in a domain have a network connection to the client and servers may be connected to other servers.

The client runs the adaptation control loop and manages the adaptation of the set of active applications, including (re)deployment of their components on the resources in its domain, seeking to maximise the utility to the user.

The applications running inside a domain may depend on services provided outside the domain. This may include both web services and shared peripherals. Discovering, selecting and binding to suitable service instances is also part of the responsibility of adaptation management, as well as replacing or removing the need for services that disappear or otherwise break the service level agreement. However, this is outside the

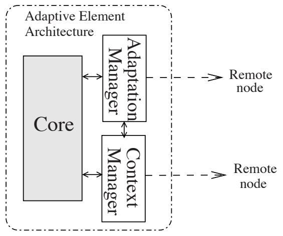


Fig. 1. MADAM Adaptive Element Architecture

scope of this paper. Here we focus on adapting the deployment of the components of the active applications on the resources available inside an adaptation domain.

3.1 MADAM Adaptive Element Architecture

Figure 3.1 shows the architecture of an instance of the MADAM middleware, which represents an adaptation control loop in one node. Its main components are the core, the adaptation manager and the context manager. The core provides platform-independent services for the management of component instances including application components, context components, and resources that are also reified through components. The management of such components involves the supporting of lyfecycle operations such as loading, unloading, binding, unbinding and setting parameters of components. This is implemented using reflective mechanisms similar to those in Fractal [5] and OpenCom [6].

The Adaptation Manager is responsible for reasoning on the impact of context changes on the application, determining when there is a need to trigger adaptation of the application, and for selecting an application variant that best fits the current context. The Context manager is responsible for managing and monitoring contexts relevant for the adaptation. It manages the contextual information available to the node, including the execution platform, with its networks and computing resources, and the physical environment information such as light and noise and user needs [7]. Furthermore, as the context manager enables distributed context information sensing and aggregation operations, additional services such as network and node discovery and context sharing are enabled.

3.2 Adaptation Approach: Property-Driven Variability

The working of the MADAM middleware is based on architectural reflection, meaning that the middleware maintains models of the running applications, with adaptation capabilities modelled explicitly in the form of variation points. It also maintains models of the user needs and the computing and communication resources available within the adaptation domain. These models are represented according to the conceptual model depicted in figure 2.

We view a software system and its context as a system of interacting entities. Entities may represent applications, instances of software components making up applications,

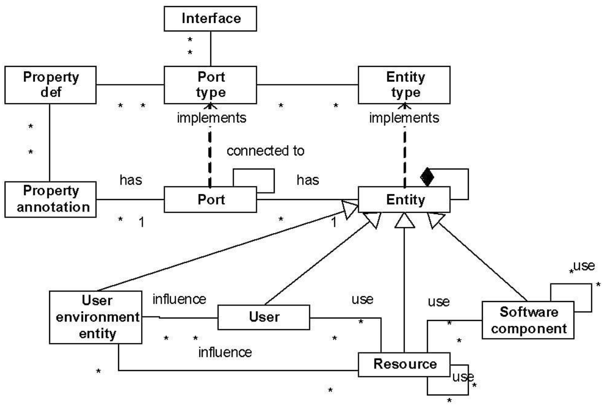


Fig. 2. Conceptual adaptation model

computing or communication resources, the user, and elements in the environment which influence the user needs. Entities interact with other entities by providing and making use of services through ports. A port represents a service offered by an entity or a service needed by an entity. Entities may be composed of smaller entities, allowing for a hierarchic structure. Distribution is modelled by dependencies of software components on resource entities representing computing devices.

To model variation points, both in the application and in the computing infrastructure, we introduce the concept of entity types. An entity type defines a class of entities with equivalent ports which may replace each other in a system.

With these concepts, we are able to model the architecture of an adaptive application as a possibly hierarchical composition of entity types, which define a class of application variants as well as a class of contexts in which these applications may operate. The latter include the computing infrastructures, on which the applications may execute. In addition to this , we need a way to enable the derivation of the software variant and its deployment on the available computing infrastructure that best fits the current user needs. Our approach is based on property annotations associated with the ports. The property annotations characterises the service provided or required by the port. For example, a property annotation might denote the response time of a service provided by an application, the latency of a communication link, the maximum latency tolerated by an application, or the noise level at the current location of the user.

Property annotations allow us to reason about how well an application variant matches its context, by comparing the properties of the services provided by the application with the properties required by the user, and the properties expressing the resource needs of the application with the property annotation describing the resources provided by the current computing infrastructure. The match to user needs is expressed in a utility function associated with each application. By default the utility function is a weighted mean of the differences between properties representing user needs and properties describing the service provided by the application, where the weights represent

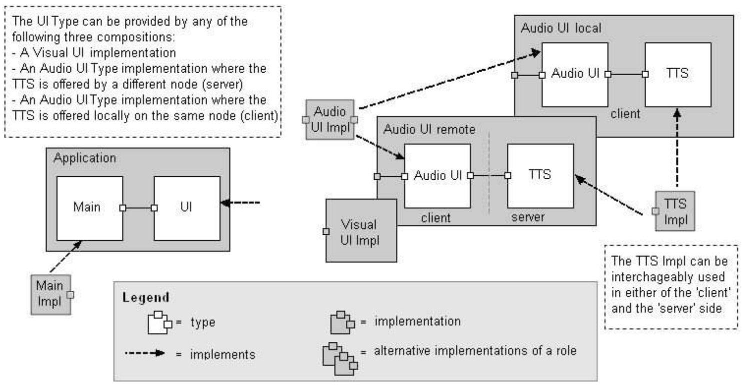

The UI Type can be provided by any of the following three compositions:
- A Visual UI implementation
- An Audio UI Type implementation where the TTS is offered by a different node (server)
- An Audio UI Type implementation where the TTS is offered locally on the same node (client)
Application
Main
UI
Main Impl
Audio UI Impl
Audio UI local
Audio UI
TTS
client
Audio UI remote
Audio UI
TTS
client
server
Visual UI Impl
TTS Impl
The TTS Impl can be interchageably used in either of the 'client' and the 'server' side
Legend
= type
= implementation
= implements
= alternative implementations of a role

Fig. 3. Application Compositions

priorities of the user [8]. However, the developer may also provide a tailored utility function for an application. The following section shows how this model is applied to our motivating example.

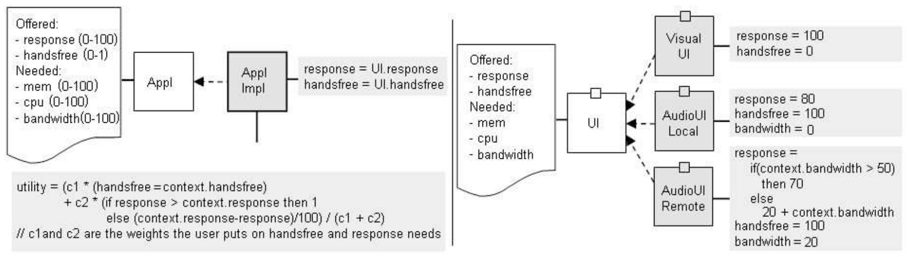
3.3 Example Revisited

The architecture of our application example consists of four components namely the main logic, the visual UI impl, the audio UI impl, and the TTS components (see figure 3). The main logic component encodes the basic control loops of the application, and is responsible for the functional implementation. This component has one dependency only: the UI type. This type can be interchangeably provided by any of the three implementations: Visual component impl, the Audio UI local, and the Audio UI remote.

By using these basic components as building blocks, the application can be configured in three different compositions (i.e. variants). The three possible compositions are as follows: (i) the main logic component is connected to the visual UI component, (ii) the main logic component is connected to the audio UI component which is itself connected to a local instance of the TTS component, and (iii), the main logic component is connected to the audio UI component which is itself connected to a remote instance of the TTS component (i.e. on a server node).

Regarding properties annotations, as it is illustrated in figure 4, our application example is modeled around two main properties: the *response* and *handsfree*. At runtime, the adaptation process tries to match the required properties to the offered ones, something which is depicted by the depicted utility function. This function is expressed as a weighted average of the user's need for handsfree functionality and quick response time. The preference among the two is controlled with the *c1* and *c2* parameters (where higher *c1* indicates greater dependency on the handsfree functionality, and greater *c2* indicates higher dependency on the response time).

Concerning the UI type, again the offered and needed properties are the same as for the application, as it is shown in the figure. In the case of using the ”Visual UI” implementation, the response and the handsfree properties offered take a fixed value.



```
utility = (c1 * (handsfree =context.handsfree)
          + c2 * (if response > context.response then 1
                  else (context.response-response)/100) / (c1 + c2)
// c1and c2 are the weights the user puts on handsfree and response needs
```

```
response =
    if(context.bandwidth > 50)
      then 70
    else
      20 + context.bandwidth
```

Fig. 4. Application variants properties

Similarly, when the "Audio UI"implementation is used where the TTS component is local, the two properties are also of fixed value. Furthermore, the required bandwidth is 0 (as no networking is involved). In the case of the "Audio UI" implementation where the TTS is remotely deployed though, the response is expressed by a function which expresses that the value depends on the bandwidth, which is reasonable as the networking affects the way the two nodes interact. In this case, the bandwidth is also set to a fixed value (i.e. 20) which is the minimum required.

4 Resource Management Framework

We understand a resource simply as a reusable entity in the system, that is employed to fulfill the resource request by a resource consumer. In our adaptation approach presented in the previous section, dependencies of a given application on resources are expressed through properties. Therefore, the resource management component is an indispensable component to the adaptation manager in order to enable deciding for the appropriate application variant and (re)deployment within an adaptation domain with respect to the user needs. To achieve that, the resource management framework should provide facilities for *discovering*, *monitoring*, *allocating and releasing*, and *configuring* resources.

4.1 Distributed Infrastructure Resource Model

A prerequisite for allowing the observation of resources is to model them so that their runtime behaviour is reified. Resources are modeled uniformly as special entities according to our conceptual model (see figure 2). In one hand this allows hiding the heterogeneity of the different resources and in the other hand, it facilitates their runtime management as every component instance.

As shown in figure 5, a resource may be atomic - e.g. network and computational node resources -, or composite - e.g. clusters of nodes -. A resource has a type and all resources of the same type provide the same set of services types which are qualified with a set of properties. These properties includes particularly QoS characteristics that represents the usage and the capacity of consumable resources. More precisely, we distinguish between three types of resources namely node, network and peripheral resources.

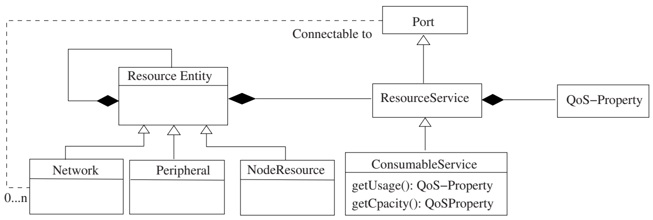


Fig. 5. Distributed infrastructure resource model

A *node resource* represents any computational node in the system that may host potential application components on behalf of the middleware using the adaptive element architecture. This resource type provides services such as memory and CPU used to execute component instances. For the particular case of adaptation domain management, one can distinguish between client (master) and server (slave) nodes. Server nodes can be viewed as grid-like computational server resources, while client nodes represent smaller (e.g. handheld) computers with fewer capabilities and additional limitations (e.g. battery and memory space) which should also be taken into account during the adaptation.

A *network resource* is fundamental to distributed infrastructures which use it to reach to and connect with different remote resources. Particularly, in our mobile setting, a handheld node may have the opportunity to use multiple network connections alternatives (WiFi, Bluetooth, GPRS, etc) between other nodes. This leads the adaptation manager to exploit and to select the appropriate network connections: for example the one with a good bandwidth, the most secure, the one that increases the availability or yet the least expensive connection. In the figure 5, a network connection may exist only between non-network resources and other resources. The implementation of such connections requires composite channel components (proxies, stubs, etc) which are not modeled in the figure. The main common services provided by a network resource are *send* and *receive* where the consumption is qualified using the *throughput* property. Depending on the underlying network technology additional properties such as those related to collision and errors sent or received and signal or noise level (wireless networks) may be considered to perform the adaptation.

A *Peripheral resource* covers the rest of the resources such as remote displays, printers and sensor devices. In our approach, these resource types are handled as application components in the sense that the adaptation manager has to discover the peripheral component services (equivalent to peripheral drivers) and then compose (i.e. connect) them with the application components following the required and provided dependencies.

In our example (see figure 4), the different needed properties of the application components properties (cpu, memory, network, etc.) are used to derive and discover all the exploitable resource types within the user adaptation domain namely the handheld device, the current media server and available networks. All these resources and the services they provide are reified through component.

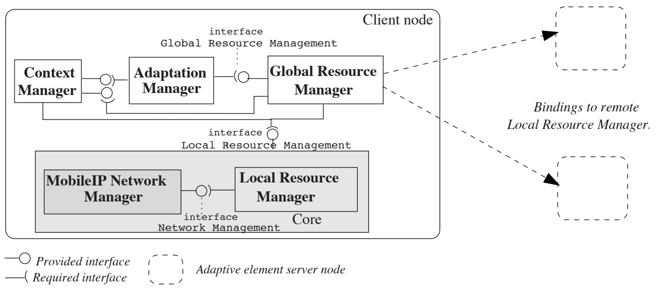


Fig. 6. Distributed resource management architecture

4.2 Distributed Resource Management Architecture

Assuming the above resource model, the resource manager provides a set of services for the exploitation and the management within an adaptation domain. Figure 6 presents the architecture of the distributed resource manager which is plugged to the adaptive element architecture presented in section 3.1. A main principle of the MADAM approach for the management of distributed resources is the separation between the Local and the Global resource manager components.

The local resource manager is part of the core middleware component. It provides a base level of manageability (access, observation and allocation) through the `Local Resource Management` interface of the local resources entities present in one node (either at the server or the client nodes). It also offers functionalities related to the network detection and connection management by interacting with the `Network Management` interface provided by the primitive MobileIP network manager component. Although this component hides the underlying heterogeneity by providing uniform resource access, its implementation depends highly on the underlying hardware resource characteristics.

The Global resource manager is only present at the client middleware node to be used by the Adaptation manager. It maintains the distributed resource model by providing a mediation interface, the `Global Resource Management`, that federates the access and the search for all the available and discoverable resources within an adaptation domain. The following sections present the different services provided by the these different components.

Resources discovery. The preliminary operation before performing any adaptation reasoning is the discovery of the available distributed resources and more generally of the available resource services. The adaptation manager uses the Context manager [7] component to search for remote resources and the Local resource manager to look for the locally available resources. The Context Manager supports plugging different alternative technologies such as TCP/IP, UPnP and Bluetooth for remotely discovering available resources and disseminating information across them. Furthermore, the proposed framework supports configuring the resource discovery to use either a pull or push policy. In the push policy, available resources and services are actively discovered just before an adaptation reasoning process is started. In contrast to this, the pull policy instructs the nodes to periodically disseminate their context information and at the same time always maintain a local model of the distributed context in the environment including resources. This configuration is possible because of the flexibility of the Context Manager which supports both of these strategies. Using either of these two alternatives depends on the state of the system, as the pull policy is more energy efficient (less communication) but less agile compared to the push policy.

The detection of the surrounding network resource types (WLAN, Bluetooth, WiFi, etc) is delegated to the MobileIP network component. Recall that these networks may potentially be used either to join a network (e.g. WLAN), or to be used as protocols for discovering other resources (e.g. Bluetooth devices). Another important discovery feature is the search for the different network paths between two given nodes. As already discussed, this leads the adaptation manager to select the different connections between the distributed component so that to maximise the user utility.

Resources allocation/releasing. Resources sharing between different adaptive applications and potentially between many adaptation domains involve mainly reserving and releasing operations. These operations are provided at both levels the local resource manager and the global resource manager components interfaces.

The Adaptation Manager calls such operation after the planning has been performed, the required resources types (including remote nodes) have been selected, and the amount of resources has been calculated. This operation invoked on the Global Resource Manager uses then the local resource managers of the different remote resources for the registration of the allocation or the releasing resulted from the new selected configuration. The Adaptation manager exploits resources information provided during the discovery, including theirs location (e.g. IP adresses), to establish remote bindings with the remote Local resource managers provided interfaces. For that, a binding framework (see deliverable D2.3 [3]) similar to the one presented in [9] has been adopted. The local resource manager maintains the different reservations per resource service and client ID. Including the client ID allows the local resource manager to release all the reserved resources for a given client in case of the client node, i.e. the mobile user, becomes unreachable.

Resources observation. The Local resource manager allows adding and removing resource listeners so that they are notified of resource usage changes that may trigger the adaptation process. The context manager is responsible for the management of the pool of context listeners, including resource listeners, and for delegating these changes to the adaptation manager.

Network management. The network management facilities are provided by the the MobileIP network component tailored for mobile mobile in IP networks. This component provides mainly operations for seeking for surrounding networks as already discussed and also the management of the connections (and disconnections) to (or from) a given network. When a given network has been selected, the adaptation manager connects to that network. Conversely, if the node was already connected to one different network type, then the adaptation manager should disconnect the node from this network before switching to the newly selected one. Note that in the current specification and implementation of the MobileIP Network Manager, a node is only connected to one network type at any given time.

Resources configuration. Our framework exhibits also configuration methods that allow setting new properties values to certain resources. In some MADAM scenarios (see D1.2 [3]), some adaptations are simply implemented by tuning resource parameters.

5 Distribution Adaptation Reasoning

When replanning is triggered, the adaptation manager uses the application model, the user needs model and the resource model to select the combination of application variants and their deployment on the available resources that best fits the current context. This involves the execution of the following pseudo algorithm:

```
v: represents an application variant from the search space V. It associates
for each component type an implementation and a node where to be deployed.
U: is the utility function that takes as input the
variant properties and the different contexts information including resources.

 1: begin
 2:   reserve and get resources // interaction with the Global resource manager
 3:     for each variant v from V
 4:       aggregate resource needs of v
 5:       if the resource needs of V can be satisfied with available resources then
 6:         compute the utility U(v)
 7:         if U(v) is better that the utility of the best variant set found so far then
 8:         keep v
 9:   release and allocate resources // interaction with the Global resource manager
10: end
```

The resource needs of a variant set are expressed as dependencies on resource types, and in an adaptation domain there may be several instances of each type. For example there may be several servers and several networks with different capacities. Therefore the adaptation manager must also decide a mapping. As long as resource needs are fixed, i.e. given as constants, this is trivial. All mappings that satisfy the resource needs are valid and have the same utility, so the first one found that satisfies the needs is selected.

However, fixed resource needs is not always an adequate model of an application variant (or component. In many cases a range given as a minimum and a maximum amount is a better model. The variant needs at least the minimum amount to execute properly, but if more resources are available the provided Qos will improve until the maximum amount. Allocating more resources than the maximum amount does not improve the provided QoS. The MADAM application modelling support such open specification of resource needs.

In the presence of open resource needs, the utility of a variant set i) depends on how its components are deployed on alternative nodes of the same type, ii) on how remote connections are mapped on available networks, and iii) how resources are distributed between the components deployed on each node and the connections mapped on each network.

Since on the class of computing infrastructures that we are targeting, cpu and network resources tend to be distributed between competing programs on a fair share basis and memory on a first come first serve basis, all outside our control, we have to base the computation of the utility on an educated guess of how resources will actually be allocated.

6 Implementation Status

The resource and the adaptation framework presented in this paper have been implemented as part of the MADAM adaptation middleware. The MADAM middleware is programmed in Java and runs both on Windows XP on ordinary PCs and on Windows Mobile on HP iPAQ handheld computers. The monitoring and control of network resources (i.e. the MobileIP Network Manager component) is based on the Birdstep[2] Mobile IP product which supports most of the existing networks. The middleware is being used in the development of two pilot applications by the industrial partners of Madam.

The Middleware can be started in either master or slave mode. Slave nodes basically only manage resources and monitor other context information and perform reconfiguration operations in response to reconfiguration requests from a master. Adaptation domains are formed by nodes sending regular multicast messages informing potential neighbours about their presence. This limits an adaptation domain to a local area network. The communication between the master and its slaves is based on RMI.

The adaptation scenario used in the example is present in both pilots, along with a number of other scenarios exercising both distribution and other forms of adaptation. A previous version of the middleware, without support for distribution adaptation but built on the same adaptation approach, has been used successfully in several pilot applications and confirms that the general approach is feasable and quite satisfactory (see deliverables D5.2 and D6.2 [3]).

7 Discussion and Related Work

What makes our work different is primarly the scope of the targeted adaptation in a mobile environment and the variability-based approach used to implement these adaptations. As a result, the proposed resource management framework is tailored towards offering specific functionalities to enable distributed adaptation planning in the MADAM middleware. In addition, the proposed architetcure is flexible and configurable. Firstly, the resource management is not bound to a particular resource type but deals with arbitrary resource types. Indeed the presented resource model is uniform and generic, implying that there is no inherent limitation in the resource model with respect to the range of resource types the resource model can represent. The properties which characterise these resource types enable the adaptation reasoning to affect and compose resource

[2] `www.birdstep.com`

types to a given application components so that the required behaviour is preserved. Secondly, the modularity of the resource management framework makes it configurable in the sense that its components may be deployed and configured to handle different requirements. For example, when there is no requirment for distribution (standalone mode), the resource management framework can be configured with the Local Resource Manager only, as in this case there is no need for the Global Resource Manager and the MobileIP Network Manager. In this configuration, the Adaptation Manager is able to locally adapt the application without involving distributed third parties. Finally, another token of flexibility of our framework is that it can be configured to support both push and pull resource discovery strategies.

Within the context of mobile environment, several systems and middlewares have been proposed to target the management of the adaptiveness of mobile applications. However, from our knowledge, most of them either are centralised or do not provide an explicit architecture of the resource management eventhough they support distribution (e.g. Aura [10]). For example, [11] proposes a resource model that is the basis of a framework for the development and deployment of adaptable applications. This resource model is used to model and declare the resources required by an application and the ones supplied by the hosting environment. To reason on the goodness of a resource set allocated to satisfy a given adaptation a utility function is used as in our approach. While this work target similar adaptations, the proposed model does not support distribution and does not address the management concern. Also, application resources needs are specified for each possible adaptation. This constitues another limitation of this model to be viable in our work since compositional and planning-based adaptation may lead to a huge number of possible adaptations.

In the context of Grid computing, many active researches have focused on designing resource management architectures such as Globus [12] and GridKit [13]. Theirs approach exhibits some similarities with ours in the sense that they distinguish between the local and the global resource managers for the management of distributed resources. Furthermore, from an architectural point of view, our framework has similarities with the GridKit architecture [13] which also proposes a flexible and configurable framework. However, all these systems do not target and consider adaptation types related to mobile applications such as those related to the resource limitations of handheld device and the support of multiple network connection alternatives between different nodes.

It is to be clarified that the computational complexity of distributed (re)deployment is not addressed in this paper. It is well known that this problem is NP-Hard [14] and therefore scalability is a serious concern. However, so far, our experience indicates that the mobile applications and environments that we target are sufficiently constrained both in term of the number of software variation points and the number of nodes and adaptations within an adaptation domain, that acceptable performance can be achieved. Indeed, for example the complexity – i.e. the adaptation reasoning – of our pilot application still inside what we found could be handled in our experiments with the previous centralised version (1000 variants gave adaptation times around 1 s). However, the "heartbeat" messages and the RMI calls for communicating between the master and the slaves seems to be expensive and particularly from the handheld device side. Therefore, further code optimisations and experiments are needed to improve the current implementation. Furthermore, we are also considering other technologies than RMI for new experiments.

8 Conclusion

In this paper we have discussed a general approach for distributed adaptation and subsequently a distributed resource model and management tailored for deployment of adaptive services in a mobile environment. Furthermore, we have presented middleware level mechanisms neccesary for maintaining an up-to-date model of the resources available in the run-time environment of a mobile device. The resulting resource framework has been realized as part of the MADAM planning-based middleware. The framework is configurable and extensible and can be customised and tailored to specific needs through support for middleware configuration. Furthermore, this framework identifies and covers new functionalities and features related to mobile computing which are not common and not covered in the classical grid-like resource management. The use of the resource management in adaptation planning was demonstrated through a real adaptive application that has been implemented on top of the MADAM middleware.

As part of the ongoing projects, two research directions are considered. Firstly, we plan to extend our approach to cover adaptation in ubiquitous computing environments. Secondly, we plan to study the possibilities of projecting and porting the proposed adaptation approach into the context of service oriented computing. This leads us to address the problem of decentralised planning in the presence of many autonomous adaptation domains. This will most likely also require the consideration of more elaborate resource management features that handle complex planning approaches such as those based on market-based and learning automata mechanisms.

Acknowledgements

This work is part of the MADAM project and the authors would like to thank all involved partners.

References

1. Amundsen, S.L., Lund, K., Eliassen, F.: Utilising alternative application configurations in context- and QoS-aware mobile middleware. In: Eliassen, F., Montresor, A. (eds.) DAIS 2006. LNCS, vol. 4025, pp. 228–241. Springer, Heidelberg (2006)
2. Poladian, V., Sousa, J., Garlan, D., Shaw, M.: Dynamic configuration of resource-aware services. In: Proceedings of the 26th International Conference on Software Engineering (ICSE) (2004)
3. Madam Consortium: Mobility and ADaptation enAbling Middleware. Delivrable are open here `http://www.ist-madam.org/consortium.html`
4. Floch, J., Hallsteinsen, S., Stav, E., Eliassen, F., Lund, K., Gjrven, E.: Beyond design time: using architecture models for runtime adaptability. IEEE Software (2006)
5. Bruneton, E., Coupaye, T., Leclercq, M., Quema, V., Stefani, J.B.: The fractal component model and its support in java: Experiences with auto-adaptive and reconfigurable systems. Softw. Pract. Exper. 36(1112), 1257–1284 (2006)

6. Coulson, G., Blair, G., Grace, P., Joolia, A., Lee, K., Ueyama, J.: A component model for building systems software. In: Proceedings of IASTED Software Engineering and Applications (SEA'04) Cambridge, MA, USA
7. Mikalsen, M., Paspallis, N.J., Floch, E.S., Papadopoulos, G.A., Ruiz, P.A.: Putting context in context: The role and design of context management in a mobility and adaptation enabling middleware. In: 7th International Conference on Mobile Data Management (MDM'06), Nara, Japan, IEEE Computer, Washington, DC (2006)
8. Alia, M., Eide, V.S.W., Paspallis, N., Eliassen, F., Hallsteinsen, S., Papadopoulos, G.A.: A utility-based adaptivity model for mobile applications. In: The IEEE International Symposium on Ubisafe Computing (UbiSafe07), IEEE Computer Society Press, Washington, DC (2007)
9. Parlavantzas, C.G., Blair, G.: An extensible binding framework for component-based middleware. In: Proceedings of 7th international conference on enterprise distributed objects computing, IEEE computer society, New York (2003)
10. Sousa, J., Garlan, D.: Aura: An architectural framework for user mobility in ubiquitous computing environments (2002)
11. Mancinelli, F., Inverardi, P.: A resource model for adaptable applications. In: SEAMS '06. Proceedings of the 2006 international workshop on Self-adaptation and self-managing systems, pp. 9–15. ACM Press, New York (2006)
12. The Globus Project : Resource management: The globus perspective, presentation at globusword, available at `http://www.globus.org/` (2003)
13. Cai, W., Coulson, G., Grace, P., Blair, G.S., Mathy, L., Yeung, W.K.: The gridkit distributed resource management framework. In: EGC, pp. 786–795 (2005)
14. Musunoori, S.B., Horn, G., Eliassen, F., Alia, M.: On the challenge of allocating service based applications in a grid environment. In: Proceedings of the International Conference on Autonomic and Autonomous Systems, vol. 43, IEEE Computer Society Press, Los Alamitos (2006)