Handling Multiple Levels of Requirements for Middleware-Supported Adaptive Systems

Pete Sawyer, Nelly Bencomo, Paul Grace, and Gordon Blair
Technical Report (COMP 001-2007), Lancaster University
Computing Department, Lancaster University, Lancaster, UK
{sawyer, nelly, gracep, gordon}@comp.lancs.ac.uk

Abstract

Adaptability is emerging as a crucial enabling capability for many applications, particularly those deployed in dynamically changing environments such as environment monitoring, disaster management, and military systems. One of the challenges that these pose to RE is that of complexity and how to handle the requirements arising from different states of the environment, and the requirements for coping when the environment changes. One approach to handling this complexity at the architectural level is to augment middleware systems with adaptive capabilities. This paper examines how adaptive middleware can be exploited by analysts handling requirements for adaptive systems. Here, requirements for adaptability, and the associated requirements for identifying when and how to adapt are allocated to the middleware. We describe how this is achieved in the Gridkit middleware that has been developed to support adaptive grid applications. Gridkit exploits a set of frameworks, each responsible for different types of middleware behaviour. This mechanism provides the basic capability for adaptation, while adaptability requirements are encoded as rules that are consulted at run-time when a change in the underlying environment is detected.

1. Introduction

Berry et al [1] have argued that, to support dynamic adaptive systems (henceforth called simply adaptive systems), four levels of requirements engineering (RE) are needed. These range from the most abstract, level 4, which is essentially the identification of requirements for mechanisms that permit systems to adapt, to the most concrete, level 1, which is concerned with the requirements for a system to operate in a particular context. Level 1 RE is essentially 'traditional' RE. What characterises adaptive systems is that they are capable of adapting to different level 1 requirements imposed by a range of contexts, adapting dynamically as the context changes. Level 2 is performed not by human analysts but by the system itself. It is concerned with how adaptive systems detect changes to their context by monitoring the extent to which the level 1 requirements are being satisfied at run-time so that dynamic adaptation can be performed accordingly. Level 3 is concerned with how analysts identify the requirements for adaptation that in turn enable the system to support a range of level 1 requirements.

In Berry's model, levels 3 and 4 may be thought of as dealing with meta-requirements. The extent to which level 2 is really RE or really the monitoring and adaptation mechanisms used to satisfy the level 3 meta-requirements is arguable. However, it is clear that adaptive systems need to have some internal model of their level 1 requirements in order that level 1 requirements satisfaction monitoring and system adaptation can be performed.

Adaptive systems are inevitably complex. This complexity requires support in the system architecture. The use of a middleware substrate [2] is one architectural solution that helps applications adapt at design and deployment time by insulating developers from the specifics of different target operating environments and network protocols. By augmenting middleware with reflective capabilities, run-time adaptation is supportable too. This helps applications cope dynamically with changes in the run-time environment such as a resource becoming unavailable.

The development of such next-generation middleware [3][4] is crucial for the development of large-scale distributed, heterogeneous adaptive systems and ultimately for the feasibility of fully *autonomic* systems [5]. Without the support of middleware capable of satisfying at least some of the crucial requirements for adaptation, the cost and complexity of bespoke adaptive system development will be too great. The development of next-generation reflective, adaptive middleware systems can therefore be seen to be founded on level 3 RE with the middleware run-time providing what Berry et al. identify as level 2 RE. This inevitably involves the teams developing adaptive middleware in level 4 RE and the development of associated level 2 mechanisms.

In this paper we use a case study involving an advanced middleware system called Gridkit [6][7] to explore Berry et al.s' ideas about levels of RE in adaptive systems. Our case study shows that current research in advanced middleware systems recognizes the need for support for the explicit specification of adaptive requirements. In particular, we show that Gridkit uses a policy mechanism based on rules for configuration and adaptation that maps well onto level 1 and level 3 RE, and that the Gridkit run-time mechanisms based on event registration and notification implement level 2. Our paper is hence part validation of the paper by Berry at al., part case study and part investigation of interaction between requirements and system architecture.

While our paper is about how adaptive middleware impacts on the way that RE is done for applications that use the middleware, other authors have addressed the requirements of middleware for supporting adaptive applications [8][9][10].

In the rest of this paper we present a brief overview of adaptive systems, focusing on a particular class of adaptive system that is typically distributed, heterogeneous and subject to change in its environment at run-time. We then examine one application as an exemplar of such systems, a forest fire-fighting scenario. We then examine the architecture of Gridkit and go on to look at how adaptive requirements are formulated from the forest fire fighting scenario and how Gridkit supports their implementation. We identify some key lessons for RE in middleware-supported adaptive systems and present a brief description of our on-going research for specifying middleware families.

2. Adaptive systems

Adaptive systems are characterised by their ability to adapt at run-time in response to change in the environment or the context in which they operate. Adaptive systems span a wide range of applications from systems whose user interfaces are capable of adapting to changes in user behaviour [11] to the yet-to-be-realised vision of fully autonomic systems that are self aware, self configuring, self optimizing, self healing, self protecting, context aware, open and anticipatory [5].

While fully autonomic systems remain a long-term challenge for computer science, an emergent class of systems that are distributed, heterogeneous and embedded within a dynamically changing environment is demanding some of the same adaptive requirements. They may comprise networks of heterogeneous devices that encompass a range of capabilities and resource availability and which interoperate to provide some required set of services. Such applications are typically required to adapt by (re-) configuring themselves in order to, for example, allow mobile devices to exploit ad-hoc networks and cope with the associated variability in resource availability. Self-configurability needs to be informed by self-awareness and, in the case of mobile applications, by context awareness too.

A scenario for forest fire fighting illustrates this well [12]. There are two user roles involved: *controllers* and *fire fighters.* Controllers manage the operation: they move fire fighters, issue commands, decide where to deploy fire sensors, and investigate real-time simulations that predict the spread of the fire. Fire fighters deploy fire-fighting equipment and are coordinated by the controllers so that they are deployed where they can be most effectively and safely used. Fire fighters also deploy sensors to collect data about wind speed and direction needed by the controllers' simulation models and air temperature data to indicate the extent and location of the fire.

Forests typically lack power and communications infrastructures. These constraints impose a number of interesting requirements for the fire-fighting problem. Fire fighters need to be in communication with each other and their controllers yet must be mobile.

Figure 1 illustrates the scenario. Fire fighters are equipped with mobile wireless communication devices. As they are deployed at the scene of the fire, they form ad-hoc connections between themselves, sensors and on-site controllers. By contrast, an infrastructure network connects all controllers. In contrast to controllers located in permanent control centres, some controllers are deployed close to the location of the fire in command vehicles. These on-site controllers are connected to the infrastructure network via, for example, satellite, GPRS, or Wireless LAN technologies.

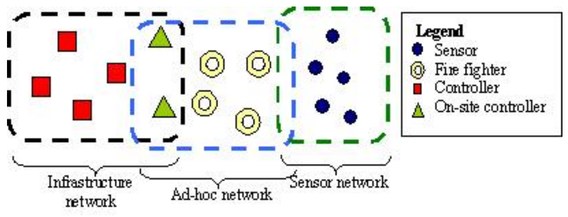

Figure 1. A forest fire fighting scenario.

The need for adaptability comes from several factors but the main ones are the mobility of the fire fighters and the sensors. Hence, the system must adapt to maintain communications between a fire fighter and the controllers as the fire fighter roams between parts of the forest served by the fixed infrastructure network and the ad-hoc network. Similarly, new sensors may be deployed so the system must be able to permit the dynamic addition to the set of sensors. Sensors may also be removed from this set as their power supplies fail or they become damaged. Finally, the quality of service provided by the different networks and, potentially, healthy versus failing sensors must be handled adaptively.

Our work posits the idea that there must be traceability from the requirements of the application and the implementation of the adaptive behaviour in such systems. The same might be said for any class of system. However, given that one of the motivations for using middleware to provide the adaptability is to reduce complexity, the adaptive behaviour as configured in the middleware must have a close, one-to-one correspondence with the adaptive requirements. This is to avoid the problem that caused early attempts to support adaptive systems to fail: "It is unrealistic to expect an adaptation framework using a "black box" approach to its adaptation intelligence to perform adequately in a generalized manner" [8].

3. Requirements for the forest fire fighting application

We will focus on one level 1 requirement arising from the forest fire fighting scenario:

1. Fire fighters must be in communication with controllers at all times.

Given the context, the range of solution options and available technology, many system requirements may be derived from this user-level requirement. One might be:

1.1 A group communication service shall enable controllers to send instructions to fire fighters.
Controllers need to be able to broadcast instructions to groups of controllers.

Although it's derived from a user requirement, requirement 1.1 is still a level 1 requirement. It is specific to the forest fire fighting problem and the solution being proposed by the system requirements. However, it leads to a requirement that the system must be capable of adapting dynamically to cope with mobility of fire fighters. To cope with this, the system must be able to sense changes to its environment and reconfigure itself dynamically.

Given the physical and technological constraints of the forest fire fighting scenario illustrated in figure 1, the implications of requirement 1.1 require much further investigation. Among the results of this, however, is the identification of associated requirements for adaptation – level 3 requirements. For example, account must be taken of the fact that fire fighters may roam between the region around the command vehicle(s) served by the fixed wireless network and that served by the ad-hoc network. Essentially, roaming fire fighters will stray beyond range of the fixed network and their communications devices will have to switch to the ad-hoc network. Requirement 1.1.1. is partly derived from 1.1 and in turn derives requirements 1.1.1.1 and 1.1.1.2 that represent, in simplified form[1], a key level 3 requirement for the system to adapt dynamically to changes to the network.

1.1.1 The network connecting fire fighters and controllers shall be transparent to both classes of user.
1.1.1.1 When a fire fighter moves beyond range of the fixed network, they shall be automatically connected to the ad-hoc network.
1.1.1.2 When a fire fighter moves within range of the fixed network, they shall be automatically connected to the fixed network.

These are generic, high-level requirements that are common to many classes of adaptive systems. Because of this it makes little sense to engineer in every kind of adaptive capability on a per-application basis. Rather, a new generation of adaptive middleware systems such as GridKit[6][7], or the Runes middleware [13] are being developed that allow application developers to

[1] For example, we assume that in the forest fire fighting scenario, there is only one fixed network and one ad-hoc network. We also assume that there will be a gateway between the ad-hoc and fixed network.

exploit their in-built adaptive capabilities. Among other requirements on the Runes middleware, the Runes requirement and constraint analysis document [14] identifies the following requirements:

"*Requirement 52*
The middleware should support dynamic reconfiguration.
The Runes middleware system should be able to be reconfigured dynamically in a fine-grained way. This allows the middleware system to adapt as a result of context, topology, mobility or resource availability changes. …

Requirement 54
There should be support for QoS adaptation.
The middleware system should be able to monitor the quality of resources and adapt, depending on application constraints…."

In terms of Berry et al. [1], these requirements are at level 3. As expressed here, they are rather general and the extent to which they can be satisfied will be constrained by the mechanisms and capabilities of the technologies used to implement the middleware. Supporting a specific application such as that outlined by the forest fire-fighting scenario therefore needs these high-level requirements to be refined in terms of the application. Just as system requirements could be derived from the level 1 requirements of the forest fire-fighting scenario, so the generic level 3 adaptability requirements need to derive requirements on how the middleware is configured.

4. Gridkit architecture

GridKit is a grid middleware technology and is built using the OpenCOM [15]component model. One of the key features of OpenCOM is that it uses a set of in-built reflective meta-models. These form the basis of the introspection capability that is a pre-requisite for adaptive functionality in middleware systems. For example, at run-time, Gridkit is able to use OpenCOM's architectural meta-model to discover the current topology of the composition of OpenCOM components from which the Gridkit implementation is constructed. An interface meta-model permits discovery of the set of interfaces defined on a component. This enables the dynamic invocation of methods defined on these interfaces, even those whose types were unknown at design time.

Gridkit can be configured to present a set of *grid services* which exploit four underlying orthogonal domains that provide generic middleware support (figure 2). These are *interaction services*, *resource discovery*, *resource management* and *grid security*. Underlying these is an *open overlays* layer that abstracts over the underlying communications support mechanisms.

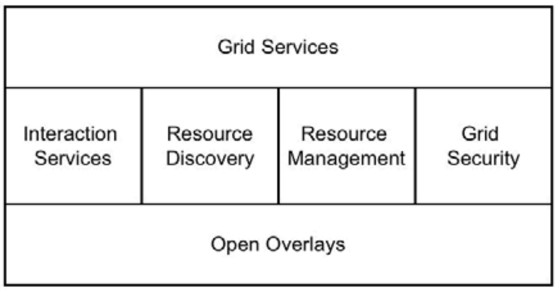

Figure 2. Gridkit architecture

Each of the middleware services and the open overlays substrate are implemented as component frameworks. We focus here on the *interaction services* framework and it's dependence on the *open overlays* framework to provide the adaptability required by the forest fire-fighting scenario.

The interaction services framework provides extensible communication services. These include support for pluggable interaction types such as publish-subscribe, multicast and streaming, as well as for QoS management. The open overlays framework provides plug-in network service capabilities. This is important for the adaptability of mobile applications which need to remain unaware of the underlying communications infrastructure. Consider the roaming of a fire fighter from a part of the forest covered by the infrastructure network, to a part served by the ad-hoc network. Here, an ad-hoc network component has to be substituted to maintain connection so they can continue to send, receive and relay information to/from controllers and sensors. However, the middleware needs to be adaptive in order to maintain connection by plugging and unplugging the network components.

Frameworks effectively serve as the domains of adaptation in Gridkit. Their adaptive behaviour is policy-driven and defined by sets of rules. Adaptation rules are scoped to individual frameworks so an instance of the interaction framework will have one set of rules, while an instance of the overlay framework will use a different set of rules. These rules define how Gridkit satisfies a requirement in a given environmental context, and also how Gridkit adapts to changed environmental context. Hence, a set of rules define how the interaction services framework delivers communications to a fire-fighter when close enough to the command vehicle to connect to its fixed wireless network, while another set of rules define how communications are delivered when out of range and relying on the ad-hoc network. A further set of rules defines how the transition is made between these two modes of communication as the fire fighter roams beyond range of the fixed network. In RE terms, the rules that specify behaviour in different contexts satisfy level 1 requirements. The rules that specify how the Gridkit middleware adapts when the context changes satisfy level 3 requirements.

5. Three levels of RE supported by Gridkit

The basic mechanisms needed to satisfy requirement 1.1 are provided by the interaction and open overlays frameworks, but we need to define the rules that specify precisely the level 1 requirements that group communication service that will be provided. We need to do the same for requirement 1.1.1 and it's implied requirement that the system will adapt dynamically to ensure that these services survive changes to fire fighters' physical contexts.

The interaction framework has to be configured to satisfy the level 1 requirements according to context. In the case of requirement 1.1, these contexts may be defined in terms of the capabilities of the device type and the communications network. This can be specified using the following rules which we have paraphrased from their machine-readable XML representations:

R1	Provide an Internet scale Application level Multicast (ALM) service for immobile devices connected to a fixed network connection. *Intended for controllers using PCs located in command vehicles.*
R2	Provide a P2P multicast overlay for mobile nodes connected to fixed wireless networks (e.g. 802.11b) to handle in and out of range disconnection. *Intended for fire fighters who are within range of the fixed wireless network located at the command vehicle but, because of their potential to roam beyond range, the ALM service used in R1 cannot be used.*
R3	For devices connected to an ad-hoc network, provide an epidemic style multicast service based upon a standard flooding approach using the broadcast channel e.g. In 802.11 in ad-hoc mode. *Intended for fire fighters deployed beyond range of the command vehicle fixed network.*

These rules represent the specification of policies. Keeney and Cahill [8] note that "Policy specifications maintain a very clean separation of concerns between adaptations available, the decision process that determines when these adaptations are performed and the adaptation mechanism itself". The decision process used in Gridkit is discussed below. The adaptation mechanism is discussed in greater depth in [6].

Figure 3 shows how rules R1 to R3 are applied. Here, Controller C1 uses their PC to communicate with fire fighters F1 to F4. C1's PC is fixed and connected to the command vehicle's fixed network so the Gridkit interaction framework running on their PC is configured using rule R1.

Despite being connected to the command vehicle's fixed wireless network, the mobile devices of F1 and F4 are configured with R2 to provide a peer-to-peer multicast overlay network. Hence, if (say) F4 was to roam beyond range of the command vehicle's fixed network into the zone served by the ad-hoc network formed by F2 and F3's devices, peer-to-peer multicast would still be available even though the underlying network had changed.

Fire fighters F2 and F3 are deployed beyond range of the command vehicle's fixed network so their devices form an ad-hoc network. Like F1 and F2, they use a peer-to-peer multicast overlay network. However, because of the ad-hoc network has different characteristics to the fixed network exploited by F1 and F4, they have to use a different multicast mechanism to propagate messages. Hence they use R3 rather than R2.

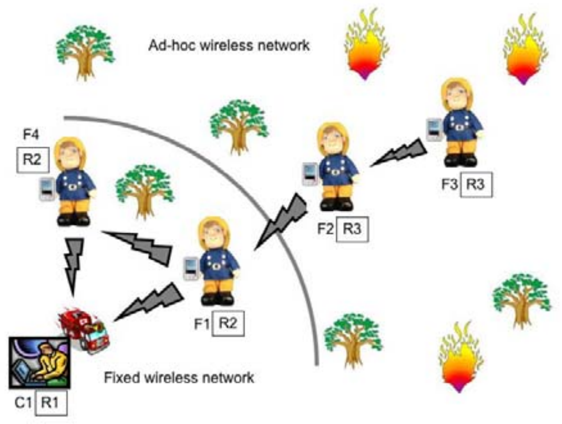

Figure 3. Deployment of configuration rules in fire fighting scenario

Rules R1 to R3 are essentially static in that they specify how level 1 requirements are met under different contextual circumstances. These need to be complemented by rules that satisfy the level 3 requirement for the system to adapt dynamically as the network changes. In terms specific to the application, the system must adapt as the fire fighter roams between zones served by the two network technologies. This is expressed by the rules R4 and R5:

R4	When the network changes from fixed to ad-hoc, replace R2 with R3.
R5	When the network changes from ad-hoc to fixed, replace R3 with R2.

Notice that R4 and R5 express only a subset of the adaptive behaviour necessary to satisfy requirement 1.1.1. insofar as they ensure that the appropriate application-level protocols needed to provide group communications (as expressed by rules R2 and R3) are used depending on the network type to which a device is connected. However, requirements 1.1.1.1 and 1.1.1.2 that specify automatic switching between network types aren't addressed by R4 and R5.

The mechanism that enables satisfaction of 1.1.1.1 and 1.1.1.2 exploits the fact that network change is one of a set of generic events that the Gridkit middleware is able to detect at run time. A Gridkit framework registers with the Gridkit *context engine* to receive notification of event types. In the forest fire-fighting scenario, the open overlays framework registers to receive notification of network changes and when a change is detected, the run-time system enacts what Berry et al. characterise as level 2 RE by invoking R4 or R5.

The rules are encoded as XML and consulted whenever either an event is triggered by a change in the environment or an application level request (such as to collect data from a sensor) is received. This separation of policy rules and event mechanism conforms to the fundamental requirements for support for adaptive mobile applications identified by Efstratiou et al. that "future systems should adopt an architecture in which mechanisms and polices are decoupled" [9].

The OpenCOM architectural meta-model is used by Gridkit's context engine to maintain a model of the system configuration needed by a framework to guide the dynamic reconfiguration. This dynamic reconfiguration is achieved by instantiating and connecting/disconnecting OpenCOM components. The OpenCOM interface meta-model permits the run-time discovery of component interfaces. Because bindings to discovered interfaces is done at run-time, a Meta-Object Protocol (MOP) is used to express contracts that must hold for a legal binding between the two parties to an interface. A run-time interception mechanism enforces these contracts which are also expressed as rules in XML.

6. Key lessons

In table 1 we summarise each of the 4 levels with examples of how they map onto the forest fire fighting scenario.

Table 1. The 4 levels of RE in terms of the forest fire fighting scenario

RE Level	Example
1. RE performed by analysts for each context-dependent state of the adaptive system. Berry et al. consider this as sets of requirements, each specifying a program for one of the contexts.	User requirement that controllers be able to communicate with fire fighters, and derived requirements for which application-level protocols should be used in which contexts. These are used to formulate rules R1 to R3.
2. Adaptation performed at run-time by the system to context changes.	Events notified via the context engine driving selection of different application-level protocols as the underlying network changes. The Gridkit runtime consulting, selecting and applying rules R4 and R5 to select between rules R1 to R3 and applying the necessary reconfiguration of components.
3. RE performed by the analysts to determine the adaptive behaviour that needs to be performed and when.	Formulation of rules R4 and R5 specifying which application level protocol to use according to which networking context. Also selecting the network change event type used to trigger application of R4 and R5.
4. Research into adaptation mechanisms.	The research undertaken at Lancaster and elsewhere into reflective adaptive middleware.

Perhaps the most interesting thing to emerge from our case study is that level 3 RE is really part of level 1 in that the requirements for adaptation are derived from the user requirements. In particular, the user requirement 1.1 is common to all states of the environment. Level 1 RE only identifies requirements specific to each environmental context once analysis of 1.1 has considered the constraints arising from the available networking technologies and device capabilities. Only once these have been identified can the level 3 requirements for change be identified. At this point an attempt needs to be made to match those requirements to the adaptive capabilities of the middleware.

Of course, our case study has focused on only one user requirement in one scenario, so the generalisability of this lesson can't be demonstrated. However, the fact that middleware systems are inevitably concerned with mitigating low-level variabilities suggests that this lesson is generalisable for a wide range of adaptive systems that are middleware–dependent, if not for other kinds of adaptive systems.

In our case study, we haven't addressed the impact of non-functional requirements (NFRs) which are often cited as the motivation for adaptation. However, a subset of NFRs at least can be addressed in the same was as the level 1 functional requirements. For example, achieving an acceptable level of safety for the fire fighters is, i* [16] terms, a softgoal that is part of the motivation for requirement 1 that fire fighters must be in communication with controllers at all times. The ability to satisfice this softgoal is impacted by the ability of the underlying network to guarantee delivery of messages. Once the requirement has been decomposed to the level that the networking technologies and communications protocols have been specified, it can be addressed by specifying another rule to complement R3 (which provides only probabilistic multicast) that a *gossip* communication service be used to propagate messages among fire fighters when connected to the ad-hoc network.

A general lesson from handling requirements for adaptive systems is that adaptation adds an extra dimension to RE. In particular, if level 1 RE yields intersecting sets of requirements according to environmental context, these sets do not map cleanly onto the accepted mechanisms for separating concerns in RE such as viewpoints or use cases, etc. Adaptability can be modeled as a softgoal [17], however. However, as shown with the fire-fighting scenario, adaptation is not necessarily closely related to stakeholder intentionality. Instead, it may emerge as a system requirement derived from a combination of user requirements and technological constraints.

Aspect-oriented techniques are being explored at the component level in reflective middleware systems [18] and aspectual views of requirements [19] may provide an alternative mechanism for identifying and managing requirements for adaptability. However, there is only a weak relationship between these so-called early aspects [20] and aspects used to partition and compose software components.

7. Future work: towards specifying middleware families

Gridkit is one of a family of middleware systems that has been developed from OpenCOM, and the Runes middleware is another example. Gridkit and Runes reflect the modern view of middleware [3][4], that a set of middleware capabilities needs to be tailored to classes of problem domains that are increasingly demanding advanced functionality such as the ability to adapt dynamically. Gridkit and the set of frameworks it provides (figure 1) is tailored to applications with characteristics like the forest fire-fighting scenario. A different set of capabilities would be needed for middleware to support other domains of application, such as environmental monitoring or pervasive computing systems.

The disadvantage of this approach is the overhead in producing different middleware systems. The OpenCOM component model greatly eases this but still entails substantial work to instantiate and configure. An important strand of our current research is to investigate how this can be made more systematic by the generation of configurations of OpenCOM components as middleware systems tailored to different domains.

We are taking a model-driven engineering (MDE) [21] approach to help with this. Using UML, we have specified a set of meta-models. These meta-models allow us to model both the core middleware functionality and the reflective functionality that is common to all middleware family members regardless of their domain [22][23]. The domain addressed by these meta-models is that of reflective, adaptive middleware and represents the fundamental component-based concepts.

The syntax and semantics offered by UML were enough to model the OpenCOM concepts. However, when specifying concepts related to higher level abstractions related to domains of application, more specific modeling concepts are needed. For example, a modeling language for developing and assembling

Publish/Subscriber applications should contain concepts like publisher, subscriber, topic, and content; concepts proper of this kind of applications. We envisage developing different DSLs for different Domains: Publish/Subscribe, Mobile Computing, GRID, Multimedia, etc. The DSLs will rely on OpenCOM concepts through the meta-models. They would be used to automate or semi automate generation of code related to the configuration and deployment of the different middleware platforms.

One obvious consequence of all this is that the requirements for domains of application should directly influence the generation of domain-specific middleware through the DSLs and meta-models. In the next phase of our research we will investigate how this is impacted by the lessons learned from the case study presented above.

8. Conclusions

Reflective, adaptive middleware has been conceived to support domains of application in which systems are typically distributed, heterogeneous and subject to change to their environment at run-time. The ability to adapt to these changes is a fundamental requirement of such systems. Their complexity demands that the role of middleware is now not only the traditional one [2] of insulating developers from the specifics of different operating environments, but now includes responsibility for providing systems' adaptive behaviour.

The approach taken by Gridkit and other members of the OpenCOM family of middleware systems, is to construct middleware systems tailored to domains of application. Here, the middleware needs to be configured to the requirements of individual applications that fall within the scope of the middleware's domain of application.

Taking the 4-level model of RE for and of dynamic adaptive systems proposed by Berry et al. [1], we have applied a case study of a forest fire-fighting application and shown how the user requirements derive requirements for adaptability. We have used this to validate the 4-level model, showing the relationship between level 1 and level 3 RE. Gridkit has a focus on insulating applications from networking technologies and the capabilities of heterogeneous devices. This means that formulation of the requirements for adaptation requires that system requirements are first derived from the user requirements to the point where the constraints arising from the underpinning technologies conceived for the solution become known.

However, once this has been done, these requirements can be directly encoded as policy rules and event types which the middleware can use at run-time to enact adaptation in the way that Berry et al. conceived level 2 RE being applied. An advantage of this is that there is a clear trace from user requirements to adaptation requirements and their implementation.

A more general finding from our case study is that RE for adaptive systems poses problems for the handling of requirements. Conventional ways to separate concerns according to use cases of stakeholder types don't map well onto the requirements for different states of the environment, which appears to be a fundamental step in identifying systems' adaptability requirements.

9. References

[1] D.M. Berry, B.H.C. Cheng, J. Zhang, "The Four Levels of Requirements Engineering for and in Dynamic Adaptive Systems", *Proc. 11th International Workshop on Requirements Engineering: Foundation for Software Quality (REFSQ'05)*, 2005, Porto, Portugal.

[2]P.A. Bernstein, "Middleware: a Model for Distributed System Services", *Communications of the ACM*, 38 (2), 1996, 86-98.

[3] F. Eliassen, A. Andersen, G.S. Blair, F. Costa, G. Coulson, V. Goebel, Ø. Hansen, T. Kristensen, T. Plagemann, H.O. Rafaelsen, K.B. Saikoski, W. Yu, "Next Generation Middleware: Requirements, Architecture, and Prototypes", *Proc. 7th IEEE Workshop on Future Trends of Distributed Computing Systems (FTDCS'99)*, Cape Town, South Africa, 1999.

[4] D.C. Schmidt, R.E. Schantz, M.W. Masters, J.K. Cross, D.C. Sharp, L.P. DiPalma, "Towards Adaptive and Reflective Middleware for Network-Centric Combat Systems", *Crosstalk The Journal of Defense Software Engineering*, November 2001, 10-16.

[5] M. Parashar, S.Hariri, "Autonomic Computing: an Overview", *Proc. Unconventional Programming Paradigms, Mont St Michel*, France, 2004, 247-259.

[6] P.Grace, G. Coulson, G. Blair, L. Mathy, W.K. Yeung, W. Cai, D. Duce, C. Cooper, "GRIDKIT: Pluggable Overlay Networks for Grid Computing", *Proc. International Symposium on Distributed Objects and Applications (DOA'04)*, Larnaca, Cyprus, 2004, 25 – 29.

[7] W. Cai, P.Grace, G. Coulson, G. Blair, L. Mathy, W.K. Yeung, "The Gridkit Distributed Resource Management Framework", *Proc. European Grid Conference*, Science Park Amsterdam, The Netherlands, 2005.

[8]J, Keeney, V, Cahill, “Chisel: A Policy-Driven, Context-Aware, Dynamic Adaptation Environment”, *Proc. Fourth IEEE International Workshop on Policies for Distributed Systems and Networks (POLICY'03)*, Lake Como, Italy, 2003.

[9]C. Efstratiou, K. Cheverst, N. Davies, A. Friday, “Architectural Requirements for the Effective Support of Adaptive Mobile Applications”, *Proc. Middleware 2000*, New York, USE, 2000.

[10] C. Efstratiou, “Coordinated Adaptation for Adaptive Context-aware Applications”, *Ph.D. Thesis*, Lancaster University, Computing Department, 2004

[11] P. Langley, “Machine learning for adaptive user interfaces”, *Proc. 21st German Annual Conference on Artificial Intelligence*, Freiburg, Germany, 1997, 53-62.

[12] C. Cooper, D. Duce, M. Younas, W. Li, M. Sagar, G. Blair, G. Coulson, P. Grace, "The Open Overlays Collaborative Workspace", *Proceedings of the UK E-Science All Hands Meeting*, Nottingham, UK, August 2005

[13] P. Costa, G. Coulson, C. Mascolo, G.P. Picco, S. Zachariadis, “The RUNES Middleware: A Reconfigurable Component-based Approach to Networked Embedded Systems”, Proc. *16th Annual International Symposium on Personal Indoor and Mobile Radio Communications (PIMRC05)*, Berlin, Germany, 2005.

[14] Fp6 IP “RUNES” – *D1.2 Requirements and constraints analysis.* http://www.ist-runes/org

[15] G. Coulson, G.S. Blair, P. Grace, A. Joolia, K. Lee, J. Ueyama, "OpenCOM v2: A Component Model for Building Systems Software", *Proc. IASTED Software Engineering and Applications (SEA'04)*, Cambridge, MA, 2004.

[16] E. Yu, “Towards Modelling and Reasoning Support for Early-Phase Requirements Engineering”, *Proc. Third IEEE International Symposium on Requirements Engineering (RE’97)*, Annapolis, MD. USA, 1997.

[17] J. Castro, M. Kolp, J. Mylopoulos, “Towards requirements-driven information systems engineering: the Tropos project”, *Information Systems*, 27, 2002, 365-389.

[18] L. Bergmans, M. Aksit, “Aspects and Crosscutting in Layered Middleware Systems”, *RM2000 Workshop in Reflective Middleware*, New York, USA, 2000.

[19] J. Grundy, "Aspect-Oriented Requirements Engineering for Component-based Software Systems”, *Proc. 4th IEEE International Symposium on Requirements Engineering (RE’99)*, Limerick, Ireland, 1999,.84-91.

[20] A. Rashid, P. Sawyer, A. Moreira, J. Araujo, "Early Aspects: A Model for Aspect-Oriented Requirements Engineering”, *Proc. IEEE Joint International Conference on Requirements Engineering (RE 2002)*, Essen, Germany, 2002.

[21] S. Kent, “Model Driven Engineering”, *IFM 2002*, volume 2335 of LNCS. Springer-Verlag, 2002.

[22] N. Bencomo, G.S. Blair, G. Coulson, T. Batista, "Towards a Meta-Modelling Approach to Configurable Middleware", *Proc. 2nd ECOOP'2005 Workshop on Reflection, AOP and Meta-Data for Software Evolution*, Glasgow, Scotland, 2005.

[23] N. Bencomo, G. Blair, "Raising a Reflective Family", *Proc. Models and Aspects Handling Crosscutting Concerns in MDSD*, Glasgow, Scotland, 2005.