Ubiquitous Computing: Adaptability Requirements Supported by Middleware Platforms

Nelly Bencomo, Pete Sawyer, Paul Grace, and Gordon Blair
Computing Department, Lancaster University, Lancaster, UK
{nelly, sawyer, gracep, gordon}@comp.lancs.ac.uk

Introduction

We are increasingly surrounded by computation-empowered devices that need to be aware of changes in their environment. They need to automatically adapt by taking actions based on environmental changes to ensure the continued satisfaction of user requirements. This complexity, of how to handle the requirements arising from different states of the environment, how to cope when the environment changes to ensure that ubiquitous systems [1] fulfill their intended purpose poses a major challenge for software engineering. One approach to handling this complexity at the architectural level is to augment middleware platforms with adaptive capabilities using reflection [2,3,4]. These augmented middleware platforms allow us to avoid building large monolithic systems that try to cover all the possible events, by providing components enabled with adaptation capabilities. These components can then be configured automatically and dynamically in response to changes in context.

Our current research is concerned with how adaptive middleware can be exploited by analysts handling requirements for ubiquitous systems. The problem here is to identify the requirements for adaptability from the user requirements, and map them onto the adaptive capabilities of the middleware in a way that is traceable and verifiable.

Adaptive middleware

Requirements for systems to dynamically adapt to changes in their environment introduce substantial complexity. In general, it is uneconomic and poor engineering practice to provide ad-hoc solutions to complex problems that share commonalities encountered within particular domains. Adaptive middleware solutions such as GridKit [5,6] or [7,8,9] mitigate this complexity in a structured way for application developers by providing adaptation support within domains of adaptation.

Gridkit is an OpenCOM-based middleware [10] solution that is structured using a lightweight run-time component model. This model enables appropriate policies to be defined and configured on a wide rage of device types, and facilitates runtime reconfiguration (as required for reasons of adaptation to dynamic environments). Gridkit supports an extensible set of middleware interaction types (e.g. RPC, publish-subscribe, streaming, etc.), and handles network heterogeneity by layering itself over virtual overlay networks which it manages and transparently instantiates on demand. GridKit exploits a set of frameworks, each responsible for different types of middleware behavior. GridKit therefore 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 [5]. By the specification of different behaviors related to different adaptability requirements, the system can be adapted without changes to the application. Although GridKit is targeted at a particular domain of application, we believe that the same adaptability mechanism is capable of supporting adaptation required in other domains.

Adaptability requirements

While software architecture has provided a technology for explicitly separating concerns in adaptive applications, requirements engineering (RE) has yet to address the problem of how to deal with adaptability requirements. Our view, echoed by Berry et al. [11], is that adaptive systems introduce conceptual levels of requirements that are orthogonal to the accepted levels of, for example, user and system requirements. In particular, requirements for adaptation are concerned with understanding how a system may either make a transition between satisfying different user requirements depending on context, or continue to satisfy the same user requirements in the face of changing context. Hence, the adaptability requirements are intimately related to, and derived from, the user requirements. Yet, they represent requirements on the satisfaction of user requirements and therefore represent a kind of meta-requirement.

We propose that RE echoes the approach taken by software architecture and imposes a clear separation of concerns between application requirements and adaptability requirements. This should have the advantage of maintaining clear traceability links between user requirements at the application level and the adaptability requirements identified by analysis and refinement of the user requirements. However, this top-down approach is insufficient since the satisfiability of the derived adaptability requirements is contingent on the adaptive capabilities of the middleware.

Again, however, software architecture provides a model that can be exploited by RE. The GridKit framework, for example, provides sets of components that can be configured for different applications using policies. As noted by Keeney and Cahill [8]: “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 policies used by GridKit are rules, expressed in XML, and can be mapped cleanly onto adaptability requirements provided the requirements are developed to the appropriate level of detail and constrained by the scope of GridKit’s domains of adaptation.

Open issues

In summary, therefore: traceability can be maintained by the derivation of *adaptability requirements* from user requirements; the requirements identified as adaptability requirements are refined and verified against the capabilities of the middleware using the semantics of the policy language used to configure the middleware component frameworks; and the verified adaptability requirements are finally encoded as policy rules while the remaining *application requirements* are implemented conventionally.

While this provides a conceptual partitioning of requirements into adaptability and application requirements, there is one outstanding problem for which a solution has not yet been identified. This is that ‘traditional’ analysis methods that are (e.g.) use-case driven or viewpoint-oriented, provide ways of partitioning the requirements that are poorly suited to identifying adaptability requirements. The need for adaptation may span several use cases, for example, yet may not easily emerge as a requirement common to the uses cases it spans. It is possible to treat adaptability as a soft goal in i* [12] but even this is problematic because adaptation is not necessarily closely related to user intentionality. Investigating this problem will form the next stage of our research.

References

1. Weiser M.: *The Computer for the 21st Century*. Scientific American, 265(3), pp. 94-104, September 1991
2. Maes, P.: *Concepts and Experiments in Computational Reflection*. Proc. OOPSLA'87, Vol. 22 of ACM SIGPLAN Notices, pp147-155, ACM Press, 1987
3. Smith B.: *Reflection and Semantics in a Procedural Language.* PhD thesis, MIT Laboratory of Computer Science, 1982
4. Kon, F., Costa, F., Blair, G.S., Campbell, R.: *The Case for Reflective Middleware: Building middleware that is flexible, reconfigurable, and yet simple to use*. CACM Vol 45, No 6, 2002
5. Grace P., Coulson G., Blair G., Porter B.: *Addressing Network Heterogeneity in Pervasive Application Environments*. Proceedings of the 1st International Conference on Integrated Internet Ad-hoc and Sensor Networks (Intersense 2006), Nice, France, May 2006
6. Coulson G., Grace P., Blair G., Duce D., Cooper C., Sagar M.: *A Middleware Approach for Pervasive Grid Environments*. UK-UbiNet/ UK e-Science Programme Workshop on Ubiquitous Computing and e-Research, 22nd April 2005
7. Capra, L., W. Emmerich, W., C. Mascolo, C., *CARISMA: Context-Aware middleware System for Mobile Applications*. IEEE Transactions on Software Engineering, Vol 29, No 10, pp929-945, Nov 2003
8. Keeney J, Cahill V., *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. Efstratiou C., *Coordinated Adaptation for Adaptive Context-aware Applications*. Ph.D. Thesis, Lancaster University, Computing Department, 2004
10. Coulson, G., Blair, G.S., Grace, P., Joolia, A., Lee, K., Ueyama, J.: *OpenCOM v2: A Component Model for Building Systems Software*, Proceedings of IASTED Software Engineering and Applications (SEA'04), Cambridge, MA, ESA, Nov 2004
11. Berry D.M., Cheng B.H.C., Zhang J., *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
12. Yu E., *Towards Modelling and Reasoning Support for Early-Phase Requirements Engineering*. Proc. Third IEEE International Symposium on Requirements Engineering (RE’97), Annapolis, MD. USA, 1997