Softure: Adaptable, Reliable and Performing Software for the Future

Antonia Bertolino*, Wolfgang Emmerich,**
Paola Inverardi*, Valérie Issarny******
*CNR, Italy, **UCL, UK,
***University of L'Aquila, Italy, ****INRIA, France

Abstract. *This paper discusses the approach that will be taken by the PLASTIC project (http://www.ist-plastic.org) in order to assist the development of adaptable, reliable and performing software services for Beyond 3rd Generation networks.*

Keywords: B3G network, service-oriented architecture, middleware, validation.

1 Overview

Software in the future (Softure) will need to cope with variability, as software systems get deployed on an increasingly large diversity of computing platforms. Software will have to be usable in various environments, due to tremendous evolution in information and communication technologies. Heterogeneity of the underlying communication and computing infrastructure, mobility and continuously evolving requirements demand new software paradigms that span the entire life-cycle from development to deployment and execution. Softure must be developed in a way that facilitates both its deployment over heterogeneous networks of heterogeneous nodes, and its interaction with end users, their environment and/or other existing systems, depending on the application domain. Moreover, Softure should be reliable and meet the user's performance requirements and needs. Softure can greatly differ in nature, varying from complex and distributed software systems for highly dynamic networks of mobile nodes to embedded software systems for wireless, resource-constrained nodes. Additionally, the user-centric dimension of the new emerging applications requires Softure to be adaptive to a context that combines user-centric data (e.g., what is the information of interest for the user given his/her current situation?) and resource/computer-centric data (e.g., what is the service that can be delivered to the user given available energy?). Finally, due to its pervasiveness, Softure must be dependable, which is made more complex given the highly dynamic nature of service provision.

Supporting the development and execution of Softure systems raises numerous challenges, from elaborating languages, methods and tools for the systems' thorough design and validation in order to ensure dependability of the self-adaptive systems that are targeted, to developing supporting middleware infrastructures in order to ease the implementation and deployment of the target systems on highly heterogeneous and dynamic platforms. The next section discusses in more details the challenges that arise to support the development of dependable, adaptable Softure systems. Section 3 then describes the approach undertaken in the IST PLASTIC project for a specific instance of Softure focused on software for Beyond 3rd Generation (B3G) networks. Section 4 presents concluding remarks.

2 Research Challenges for Softure

Various abstractions for modeling behavioral adaptation of applications in response to changes in the executing environment have been proposed recently, ranging from resource aware programming to adaptive software architectures.

2.1 Developing Adaptable Applications

Proposed solutions to supporting the development of adaptable applications include approaches for the customization of the source code. For example, aspect oriented programming (AOP) for conventional computing infrastructures has gained popularity to increase programming flexibility [1]. More advanced solutions propose context-aware programming for ad hoc mobile environments [2]. Overall, the objective of these approaches is to assist the development of applications that are *generic* and can be adapted with respect to a dynamically provided context, which is in particular characterized in terms of available (hardware or software) resources, each one with its own characteristics. It is then crucial to enforce correctness of the adaptive, generic applications, which can be achieved using a declarative and deductive approach that enables the construction of generic adaptable application code and its correct adaptation with respect to a given execution context [6, 7]. Yet another approach to adaptability of software applications is presented in [3, 4], which focus on correct dynamic linking rather than on correct tailoring of generic applications. Specifically, these papers address dynamic software update using verifiable native code, such as Proof Carrying Code (PCC), to deliver correct patches that may be applied at runtime to software, according to availability requirements. The approach requires the code to be written so that it can be dynamically updated. It deals with dynamic linking in order to patch on the fly executable code, and with code verification in order to ensure that the received patch is correct with respect to some safety properties. Along this line of research, it is worthwhile mentioning the research project on Resource Aware Programming [5] ended in 2005, which focuses on functional programming and on programming in presence of bounded resources in the more confined context of embedded software. The newly started Global Computing IP project MOBIUS [38] is also of relevance although focusing on security. Still, the security attribute is taken in its full generality and accounts also for resource usage and management and for distribution and mobility. MOBIUS proposes the use of PCC techniques integrated with type theory in order to attach to (mobile) components certificates that state the security and safety properties of the dynamic behavior of the component.

Adaptation of a software system may also be addressed in a compositional way. In that direction, software architectures are very effective tools for the description and the modeling of complex systems in terms of the composition of component systems [8]. Software architectures support the description of the static and dynamic components of the system together with how they interact. Software architectures are the basis for early analysis, verification and validation of software systems. Software architectures are the earliest comprehensive system model in the software lifecycle built from requirements specification. They are increasingly part of standardized software development processes because they represent a system abstraction in which design choices relevant to the correctness of the final system are taken. Software architectures can be used for validating the final system with respect to architectural properties [39]. Moreover, software architecture modeling is also particularly useful for assessing, so-called non-functional requirements, notably dependability requirements such as security and reliability [40] and quantitative requirements like performance and scalability [41]. Therefore, much of software architecture research has been concentrated on the specification and analysis of both qualitative and quantitative system properties. The state of the art in the field of software architectures for dependable self-adaptive distributed systems is still preliminary and fragmented, focusing on a small set of the systems' attributes. Software architectures for self-adaptive systems can be found in the literature, e.g., [9-12]. However, proposed approaches still lack associated design and validation methodologies and further address very specific adaptation with respect to computer-centric context awareness. Comprehensive design and validation of self-adaptive systems require accounting for both functional and non-functional properties in front of highly dynamic environments. Some proposals in this direction are emerging for service oriented architectures [42]. In addition, adaptation applies to both the application and middleware layers, regarding both the overall distributed systems and component embedded systems.

2.2 Middleware Supporting Adaptation

As argued above, development of adaptive software systems shall be accounted for at the middleware layer, which must both adapt its behavior according to context (i.e., available computing and networking resources and application requirements) and provide relevant feedback about the underlying infrastructure to the application layer. The former requirement may be addressed using component-based middleware, which allows enforcement of the required quality of service through the integration of adequate middleware-related services [13]. However, customization of the middleware is mostly addressed at design time, including possible middleware adaptation to deal with environmental changes [14]. To effectively support the development of Softure, context-aware composition must be enabled anytime, anywhere and hence must not rely on a priori knowledge of the computing environment, and in particular of available service instances. In other words, while most existing middleware systems require application developers to specify the instances of (middleware- and application-related) services to be used in the composition of applications, this composition shall be automated with respect to the context in Softure. Further, middleware for Softure will deal with the heterogeneity of the networking and computing environment. This includes both (i) addressing the integration of available middleware that offer effective support for their target application domains, and (ii) devising novel, advanced middleware for emerging application domains and/or infrastructures. Solutions to the former issue range from providing a new middleware API that allows benefiting from the various functionalities of the integrated middleware (e.g., [15, 16]) to implementing transparent mapping from one middleware to another at the network layer (e.g., [17]). The latter issue ranges from devising new solutions to meeting traditional non-functional requirements in order to cope with the evolving computing environment (e.g., trust management to deal with privacy and security requirements in the open networking environment [18]), to introducing new middleware architectures to cope with the specifics of emerging networking infrastructure and devices (e.g., B3G networks [19], sensor networks [20]).

Many middleware systems have concentrated on abstracting the complexities of supporting context-awareness [21-23], by providing transparent communication mechanisms in a pervasive environment, and resource management that supports adaptations to the particular domain in which the infrastructure operates. The approaches on which the systems are based vary from system to system, though some key middleware paradigms are favored. For instance, for decoupling software components, supporting complex communication patterns, and allowing transparent communication between objects, the publish-subscribe (distributed event service) paradigm has been used considerably [24-25]. In general, various middleware infrastructures have been proposed since the end of the last century to support the development of context-aware systems. The i-Land project supports the collaboration of people within an environment full of ubiquitous components [26]. An infrastructure called Beach [27] supports the dynamic configuration of the components. The COAST framework [25] provides Beach with the functionality to distribute, replicate and synchronize objects. Aura [28] has been developed for wearable, handheld devices, providing for nomadic, possibly intermittent file access, resource monitoring and application-aware adaptation. Gaia [29] enables the coordination of software entities and heterogeneous networked devices by extending the typical operating system concepts to include context and other basic services. Dynamic and automated composition of middleware-related services for enforcing dependability according to the context is addressed by the CANS infrastructure [30], which allows components to be injected into the network for dynamically adapting the system to resource characteristics of end devices and network links. A similar approach is undertaken by the WSAMI environment [31], which builds on the Web services architecture and realizes on-line distributed connector customization for increased availability of services.

As outlined above, providing solutions to the development of Softure that are adaptive to the rich context raises a number of challenging issues relevant to all phases of software development. And, although useful approaches have emerged since the early 2000s, open problems remain to be addressed and solutions need be integrated into a comprehensive development platform, possibly aimed at a specific application domain and/or computing infrastructure. In that direction, the following presents an overview of the research that we will undertake as part of the IST project PLASTIC (Providing Lightweight and Adaptable Service Technology for pervasive Information and Communication) project (http://www.ist-plastic.org), which has commenced in February 2006. Specifically, the PLASTIC project focuses on assisting the development of adaptable software services for B3G distributed computing platforms, in particular enforcing dependability of services. The PLASTIC consortium brings together expertise in the target technical fields: the University partners (University of L'Aquila, UCL, and University of Lugano) have unique expertise in devising novel software engineering solutions for advanced software applications, and in particular applications for next generation, wireless networks; the national research institutes partners (INRIA and CNR-Pisatel) have a long track record in investigating solutions for supporting the development of innovative applications for next generation networks; and the industrial partners (IBM, Siemens Business Services, Telefonica, Virtual Trip, Pragmatica Technologies and 4D Soft) together provide major capabilities for the development and deployment of software services for next generation networks.

3 The PLASTIC Approach

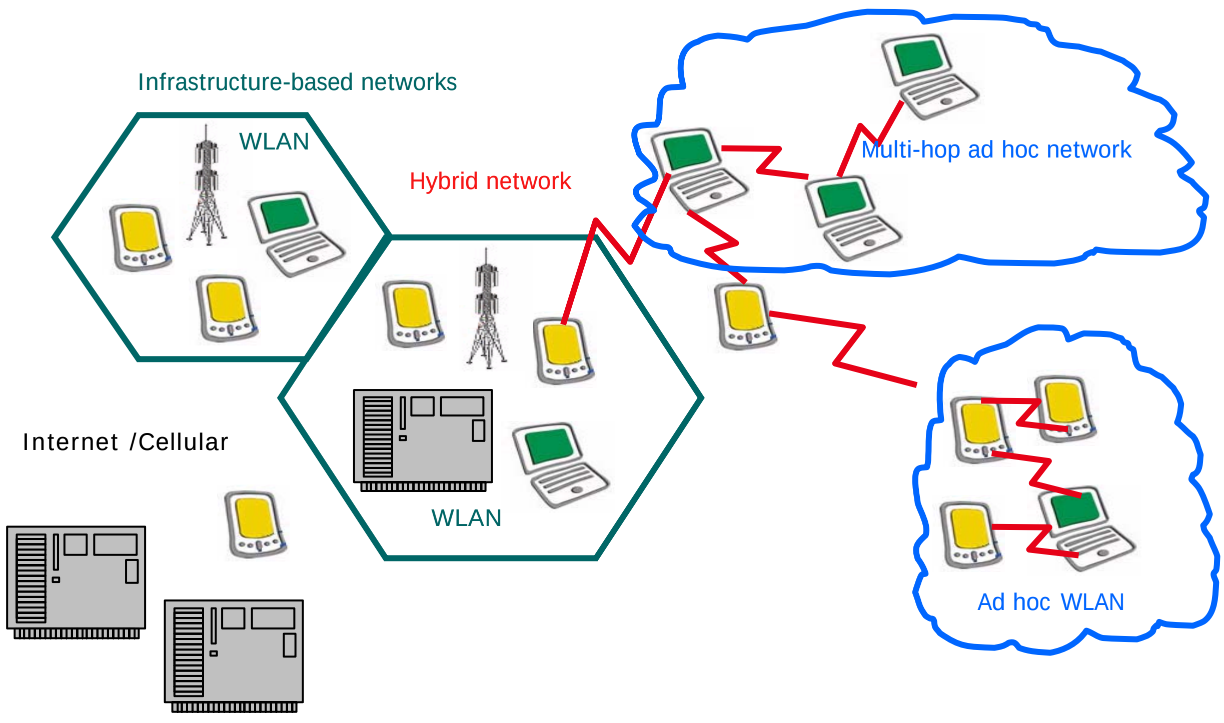

Figure 1: The B3G open wireless environment

The PLASTIC project aims to offer a comprehensive provisioning platform for software services deployed over B3G networks (see Figure 1), integrating a supporting development environment and middleware. The platform will enable dynamic adaptation of services to the environment with respect to resource availability and delivered QoS, via a development paradigm based on Service Level Agreements and resource-aware programming. The middleware will be service oriented, to enable integration and composition of heterogeneous software services from both infrastructure-based and ad hoc networks. The middleware will integrate key functions for supporting the management of adaptive services in the open wireless environment, dealing with resource awareness and dependability. The following section further discusses the PLASTIC vision, leading to introduce the supporting PLASTIC platform in Section 3.2. Sections 3.3 and 3.4 then concentrate on the two key elements of the PLASTIC platform, i.e., (i) the PLASTIC development environment enabling the comprehensive development of dependable, adaptive services, which is complemented with (ii) the PLASTIC middleware supporting the deployment and execution of adaptive software services over the heterogeneous platforms that are networked in B3G.

3.1 The PLASTIC Vision

The vision of PLASTIC is that users in the B3G era should be provided with a variety of application services exploiting the network's diversity and richness, without requiring systematic availability of an integrated network infrastructure. The success of the provided services then depends on the user perception of the delivered Quality of Service (QoS), which varies along several dimensions, including: type of service, type of user, type of access device, and type of execution network environment. In order to manage these various factors, the network's diversity and richness must be made available and be exploitable at the application layer, where the delivered services can be most suitably adapted. This demands a comprehensive software engineering approach to the provisioning of services, which encompasses the full service life cycle, from development to validation, and from deployment to execution.

The PLASTIC answer to the above needs is to offer a comprehensive platform for the creation and provisioning of lightweight, adaptable services for the open wireless environment. Various service delivery platforms have been proposed for the 2G+ to 3G cellular networks (e.g., JAIN [32], 3GPP [33] initiatives including CAMEL, OSA and IMS, PARLAY [34]). However, these platforms are focused on network-layer services. For B3G networks, there are proposals for extending the above solutions, dealing with the provision of network-layer services that can be adapted at the middleware layer [35]. At the application layer, modeling, development and deployment tools for programming, uploading and instantiating applications and code on mobile, wireless devices have been in use worldwide by many manufacturers. The major standards in this space include J2ME [36], OSGi [37] and others. Java virtual machines, lightweight messaging systems, lightweight Web Services tool kits, small-footprint databases for device controllers and programming environments have been developed (e.g., see the software available at https://secure.alphaworks.ibm.com, which will be exploited in the development of PLASTIC software tools and middleware). In addition, much research has been conducted in the areas of mobile computing frameworks, mobile grids and environments for enabling ad hoc communication and integration. The techniques, methods, tools and programming models are still evolving. However, they are primarily focused horizontally, which is to say, on a single layer of the system's infrastructure. For example, the state of the art in adaptiveness to resources and QoS, which is a key feature of mobile adaptive services, addresses individually the network, middleware and application layers.

A key challenging contribution of the PLASTIC project is to coherently manage adaptation in a vertical way across the different layers, from application to middleware to network. This will be achieved by modeling and supporting the relevant characteristics of the various heterogeneous infrastructures, so that they are made visible and manageable to the application layer through an integrated service development and execution platform. Key research points are:

- The identification of the fair tradeoffs among resources to be made visible at the application and middleware layers, and adaptation capabilities of the service;

- The ability to maintain QoS through adaptation.

The core objective of the PLASTIC project is to enable the development and deployment of cost-effective application services, both in terms of development and usage costs, regarding both financial and resource usage aspects, for B3G networks. Service development platforms for B3G networks will be effective and successful only if the services they deliver are adaptive and offer quality of service guarantees to users despite the uncontrolled open wireless environment, which will be a key focus of the PLASTIC project.

3.2 The PLASTIC Platform

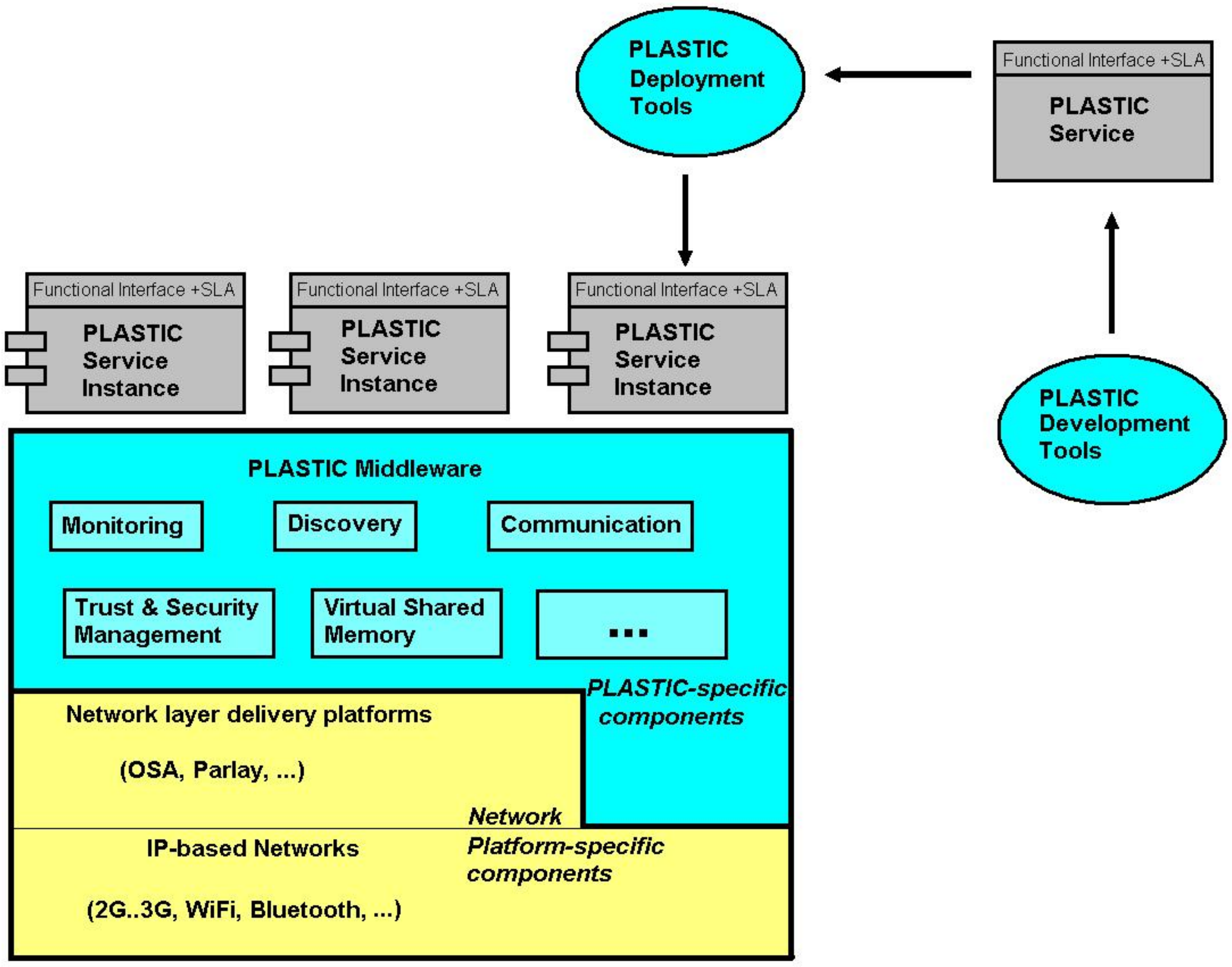
PLASTIC Deployment Tools
Functional Interface +SLA
PLASTIC Service
Functional Interface +SLA
PLASTIC Service Instance
Functional Interface +SLA
PLASTIC Service Instance
Functional Interface +SLA
PLASTIC Service Instance
PLASTIC Development Tools
PLASTIC Middleware
Monitoring
Discovery
Communication
Trust & Security Management
Virtual Shared Memory
...
PLASTIC-specific components
Network layer delivery platforms
(OSA, Parlay, ...)
Network Platform-specific components
IP-based Networks
(2G..3G, WiFi, Bluetooth, ...)

Figure 2: The PLASTIC platform

The PLASTIC platform (see Figure 2) will integrate software development methods and tools, and supporting middleware, enabling service provision in the open wireless environment. Specifically, the PLASTIC platform shall support:

- development of adaptive services for all-IP networks, i.e., development of QoS- and resource-aware, platform-independent, dependable services that adapt to the networking environment to deliver the best achievable quality of service and that may be deployed on a rich variety of devices, including wireless, resource-constrained devices;
- deployment and adaptive composition of services in the wireless environment, whether ad hoc, infrastructure-based or a combination of both, so as to realize complex and rich applications and make them available in most environments;
- run-time service management oriented toward monitoring and maintaining a quality of service that meets user expectations.

In order to fulfill the above requirements, research will be pursued in the following directions:

- Development and provisioning of robust adaptive services for the open wireless environment:
    - Service robustness will be promoted through integrated software engineering methods and tools, from design to validation. Service development will in particular build on the software engineering paradigms of service-oriented and component-based computing, i.e., an application is defined as the (possible) composition of autonomous, networked services, with an individual service being developed as a composition of components. This will allow the exploitation of existing development support oriented toward the functional robustness of applications.
    - Services will be adaptive to the environment with respect to resource availability and delivered quality of service, via a development paradigm based on Service Level Agreements (SLAs) and resource-aware programming.
- Middleware for service provisioning and composition in B3G networks:
    - The middleware will be service oriented, to enable integration and composition of heterogeneous software services, including services from both infrastructure-based and ad hoc networks.
    - The PLASTIC technology aims to be compatible with existing standards. PLASTIC will thus provide service developers with a set of platform-independent abstractions that will leverage, extend or interface with open APIs like those of OSA/PARLAY or of specific protocols. On the client side, the PLASTIC technology will allow service adaptation with respect to different access protocols (e.g., SMS, MMS, and WAP) and will provide the necessary run-time support to execute the service according to the user's expectation regarding QoS. In general, the PLASTIC middleware will allow interaction with, and exploitation of, services from the cellular network, using standard APIs defined for service platforms aimed at 2 to B3G networks.
    - The middleware will integrate key functions for supporting the management of adaptive services in the open wireless environment, dealing in particular with resource awareness, dependability, trust and security.
- Testing methods and tools to validate the dependability of mobile, adaptive services:

- Due to the mobility and strong dynamism of the considered systems, PLASTIC must provide new techniques for service evaluation and testing. In particular, dynamic adaptability is a pervasive requirement that is not adequately addressed by traditional testing methodologies. Hence, the PLASTIC platform will embody mechanisms to verify that an application, or part of it, will be able to "correctly" interact in different environments by taking advantage of the services that are available.
- QoS (particularly performance) constitutes another important characteristic of service-oriented architectures that needs to be carefully assessed. As QoS of mobile service-oriented applications is heavily influenced by communication mechanisms, the PLASTIC project will experiment with the applicability of testing methodologies for empirical QoS evaluation.

The two next sections further discuss the PLASTIC development environment and associated middleware, focusing on the requirements that we aim to address.

3.3 The PLASTIC Development Environment

Supporting the development of resource-aware and self-adapting components composing adaptable services requires focusing on the Quality of Service (QoS) properties offered by services. Although the functional properties of services are equally important for assuring the development of a competitive service, current component and service development technologies offer good solutions to achieve required levels of quality in the composition of components and services. To this respect, the PLASTIC platform will make use of consolidated technologies at the design and at the implementation level. In particular, the use of UML and MDA (Model Driven Architecture) for the design, and the use of the Java family languages for implementation will be considered.

For the management of non-functional/Quality of Service (QoS) properties offered by services, which is one of the innovative cornerstones of the PLASTIC platform, the work will rely on the specification of service level agreements (SLAs), i.e., abstract specification of the Quality of Service (QoS) properties offered by services as a compromise with the capabilities of the target platforms. In order to manage SLAs, we lay on analysis techniques that are capable of reasoning about the resources that components require in order to meet given service levels. An important objective is to devise the required notations and tools to support such designs. In order to support systematic development from the design level to the implementation of adaptable services and components, we will investigate the use of transformational approaches and rely on techniques and tools that emerge from MDA and generative programming research areas.

We will further develop a methodology for the validation of mobile adaptable component-based services from the standpoint of both functional and non-functional properties. The methodology will be supported by a test framework, part of the PLASTIC platform. Due to mobility requirement and to the strong dynamism of the considered systems, we need to study and implement new techniques for evaluation and testing. This includes the development of strategies to identify test cases aimed at validating interoperability aspects in the composition of networked services. These strategies will be built in line with the above innovative modeling and development approaches devised. In particular, dynamic adaptability is a new pervasive requirement that is not adequately addressed by traditional testing methodologies. Hence, we need to provide the PLASTIC test framework with mechanisms to verify that an application, or part of it, will be able to "correctly" interact in different environments by taking advantage of the services that are available. In addition, the same application will have to interact with newly dynamically added services at runtime. Hence we will have to devise related methods and tools to be deployed for on-line and off-line testing. We will also study the applicability of testing methodologies for empirical QoS evaluation.

3.4 The PLASTIC Middleware

The PLASTIC service-oriented middleware will support the PLASTIC development methodology for the deployment of mobile, adaptable services in beyond 3G networks. Specifically, the middleware shall be deployed over a large diversity of computing platforms, including wireless, resource-constrained devices, so that services may indeed be deployed on the mobile, wireless devices and not only accessed from them. The middleware shall further interface with the large diversity of networks composing the beyond 3G network. This includes benefiting from the functionality of latest network service platforms. Specifically, the PLASTIC middleware will be service-oriented, offering core middleware functions for the naming and discovery of networked services, and for interactions among networked services. The core middleware will further be enriched with a number of middleware-related services that are of prime importance in enabling mobile, adaptable services. Also, the middleware will allow applications to use high-level network-layer services –if and when available (e.g., location-awareness).

The core middleware supporting the PLASTIC mobile and adaptable services will build upon open standards related to service-oriented architectures and to network-layer service delivery platforms. Specifically, the PLASTIC middleware will adapt technologies related to the Web services architecture so as to enable development and deployment of PLASTIC-compliant Web services on various platforms, including wireless, resource constrained devices. Following the PLASTIC development environment for adaptable services, Web services will in particular be enriched with the specification of SLAs, further leading to related SLA-aware service discovery and access.

A key aspect of the PLASTIC middleware is to support interoperability among services deployed on devices that are heterogeneous in the software and hardware dimensions. Interoperability is addressed in PLASTIC by undertaking a service-oriented approach, which has been proven quite successful for the development of distributed applications in open environments, through the Web services architecture. PLASTIC further adopts an all-IP network view, leading to networks structured around IP domains. Services within one IP domain may then interact according to the access control policy of that domain, while interaction spanning multiple domains is made possible by dedicated bridges. The PLASTIC middleware operates over IP domains enabled by the network operator and/or ad hoc networking; PLASTIC does not offer any bridging functionality, for which various solutions may be found in the literature. The PLASTIC middleware makes available network-layer functionalities to the applications, like functions defined by service delivery platforms for 2 to 3G networks and more recently for beyond 3G networks. One key issue that arises is then which of the network-layer services should be offered to applications as is and which should be exploited at the middleware-layer and made transparent at the application-layer. For instance, the middleware may exploit the existence of multiple radio interfaces when interacting with the environment, so as to favor cost-effective interactions with respect to financial cost for the user, resource-usage and network-level quality of service. Also, the PLASTIC middleware will rely on capabilities of the underlying network for seamless mobility. The impact of network heterogeneity on the middleware and application behavior will then be addressed through the elicitation of a comprehensive semantics for service interaction (i.e., connector types offered by the PLASTIC middleware), accounting for the various network-related properties of relevance (e.g., impact of mobility management on failure and synchronization semantics) based on adequate interfacing with the network.

Service management includes service discovery, access, reservation and accounting&billing, for which we additionally have to consider mobility and security issues. Regarding service mobility, we distinguish between logical and physical mobility. As mentioned above, physical mobility relies on its handling by the underlying network, and is addressed through the definition of appropriate interaction semantics. Logical mobility is supported by enabling the downloading of appropriate software components to dynamically compose services. Context-aware discovery of networked services in the PLASTIC middleware will take into account SLAs associated with the services, prior client-side subscriptions (service advertisement and service push) and reachable network domains. In addition, since there is a large number of service discovery protocols, each aimed at a specific network, we will reuse existing solutions to service discovery protocols interoperability (e.g., [15, 17]) so that the PLASTIC middleware may build upon them. Regarding service access, the PLASTIC middleware will rely on interaction functionalities provided by the core middleware and will further support functionalities for the orchestration/choreography of distributed services. Last but not least, the PLASTIC middleware will provide functions to enforce SLAs in the dynamic network environment. This includes monitoring of resources and adapting service composition and access based on specified SLAs and evolution of resource availability.

4 Conclusion

As information and communication technologies get increasingly pervasive, diverse and rich, *Software for the future* (Softure) must be adaptive to context, i.e., must change its behavior according to the context in which the software applications are provisioned and accessed. In addition, Softure needs to be self-adaptive according to both the resource- and user-centric context, which evolves over time. The IST project PLASTIC aims at offering comprehensive development support for Softure and focuses specifically on Softure to be deployed over next generation, B3G networks.

The PLASTIC methodology embraces both service creation and service execution. Thus, the PLASTIC platform comprises software tools for the creation, development and testing of software services/components and a middleware to support service deployment and execution with respect to established Quality of Service (QoS) parameters in the B3G networking environment. Service development also includes service/components discovery and their orchestration for the creation of new service instances. In this context, development is mixed with dynamic execution, which requires dynamic adaptation and customization of generic components and thus innovative validation techniques (e.g., pre-static certification, synthesis, etc.).

The PLASTIC approach is based on component and service technologies, making services adaptable with respect to the components they integrate and the networked services with which they interact, according to the networking environment and QoS (non-functional) requirements. Adaptable service composition, from the service- to component-level, is then realized through suitable characterization of components/services. This in particular allows validating components/services with respect to interaction and coordination, including QoS requirements, and independently of the actual execution platform.

The PLASTIC platform will support the development of diverse application services exploiting the significant number and diversity of wireless resources that are (expected to be) networked, thanks to next generation, B3G networks. “Killer applications” in such a pervasive mobile environment are yet to be identified. Also, success of the PLASTIC platform depends on the achieved cost-effectiveness, regarding the cost of both service development and service usage, compared to developing services for the infrastructure-based/cellular network. The PLASTIC project will address the latter issue, and may be the former one, through the development of a number of mobile e-services in the areas of eHealth, eVoting, eLearning and eBusiness.

References

[1] CACM. Issue on Aspect Oriented Programming. Communications of the ACM. 44(10). October 2001.

[2] C. Julien and G-C. Roman. Egocentric Context-aware Programming in Ad hoc Mobile Environments. In Proceedings of SIGSOFT FSE. 2002.

[3] M. Hicks, S. Weirich, and K. Crary. Safe and Flexible Dynamic Linking of Native Code. In Proceedings of TIC 2000, LNCS 2071. 2001

[4] M. W. Hicks, J. T. Moore, and S.Nettles. Dynamic Software Updating. In Proceedings of the SIGPLAN Conference on Programming Language Design and Implementation. 2001.

[5] RAP Project. Resource Aware Programming. http://www.cs.rice.edu/~taha/RAP/.

[6] P. Inverardi, F. Mancinelli, and G. Marinelli. Adaptive Applications for Mobile Heterogeneous Devices. In Proceedings of the 22nd International Conference on Distributed Computing Systems Workshops. 2002.

[7] P. Inverardi, F. Mancinelli, and M. Nesi. A Declarative Framework for Adaptable Applications in Heterogeneous Environments. Proceedings of the 19th ACM Symposium on Applied Computing, 2004

[8] D. Garlan. Software Architecture: A Roadmap. The Future of Software Engineering, aside ICSE00, ACM Press. 2000.

[9] S.-W. Cheng *et al*. Using Architectural Style as a Basis for Self-repair. In Proceedings of the 3rd Working IEEE/IFIP Conf. on Software Architecture (WICSA 2002). 2002.

[10] E. M. Dashofy, A. van der Hoek, and R. N. Taylor. Towards Architecture-based Self-Healing Systems. In Proceedings of WOSS '02. 2002.

[11] B. Schmerl and D. Garlan. Exploiting Architectural Design Knowledge to Support Self-repairing Systems. In Proceedings of the 14th International SEKE Conference 2002.

[12] D. Garlan, S. Cheng, and B. Schmerl, Increasing System Dependability through Architecture-based Self-repair, in Architecting Dependable Systems, R. de Lemos, C. Gacek, A. Romanovsky (Eds), Springer-Verlag, 2003.

[13] V. Issarny, C. Kloukinas, and A. Zarras. Systematic Aid for Developing Middleware Architectures. Communications of the ACM, Issue on Adaptive Middleware, 45(6). 2002.

[14] G. Blair, L. Blair, V. Issarny, P. Tuma, and A. Zarras. The Role of Software Architecture in Constraining Adaptation in Component-based Middleware Platforms. In Proceedings of the ACM/IFIP International Middleware Conference. 2000.

[15] P-G. Raverdy and V. Issarny. Context-aware Service Discovery in Heterogeneous Networks. In Proceedings of the IEEE International Symposium on a World of Wireless, Mobile and Multimedia Networks (WoWMoM'2005). 2005.

[16] P. Grace, G. Blair, and S. Samuel. Middleware Awareness in Mobile Computing. In Proceedings of the 1st International ICDCS Workshop on Mobile Computing Middleware. 2003.

[17] Y-D. Bromberg and V. Issarny. INDISS: Interoperable Discovery System for Networked Services. In Proceedings of the ACM/IFIP/USENIX 6th International Middleware Conference. 2005.

[18] Proceedings of the International Conference on Trust Management. LNCS. 2003-2005.

[19] Sensor Networks. References at http://www.research.rutgers.edu/~mini/sensornetworks.html.

[20] E2R Workshop on Reconfigurable Mobile Systems and Networks Beyond 3G. http://e2r.motlabs.com/workshops/e2r-workshops. 2004.

[21] A. K. Dey and G. D. Abowd. The Context Toolkit: Aiding the Development of Context-aware Applications. In Workshop on Software Engineering for Wearable and Pervasive Computing (CHI '99). 1999.

[22] C. D. Kidd, R. J. Orr, G. D. Abowd, C. G. Atkeson, I. A. Essa, B. MacIntyre, E. Mynatt, T. E. Starner, and W. Newstetter. The Aware Home: A Living Laboratory for Ubiquitous Computing Research. In Proceedings of the 2nd International Workshop on Cooperative Buildings.1999.

[23] L. Capra, W. Emmerich, and C. Mascolo. CARISMA: Context-Aware Reflective mIddleware System for Mobile Applications. IEEE TSE, 29(10). 2003.

[24] A. Carzaniga, D. S. Rosenblum, and A. L. Wolf. Design and Evaluation of a Wide-Area Event Notification Service, ACM Transactions on Computer Systems, 19(3). 2001.

[25] C. Schuckmann, L. Kirchner, J. Schümmer, and J. Haake. Designing Object-oriented Synchronous Groupware with COAST. In Proceedings of the ACM Conference on Computer Supported Cooperative Work. 1996.

[26] N. A. Streitz, J. Geiÿler, T. Holmer, S. Konomi, C. Müller-Tomfelde, W. Reischl, P. Rexroth, P. Seitz, and R. Steinmetz. i-Land: An Interactive Landscape for Creativity and Innovation. In Proceedings of the ACM Conference on Human Factors in Computing Systems. 1999.

[27] P. Tandler. Software Infrastructure for Ubiquitous Computing Environments: Supporting

Synchronous Collaboration with Heterogeneous Devices. In Proceedings of UbiComp 2001: Ubiquitous Computing, LNCS 2201. 2001.

[28] A. Smailagic, P. Steenkiste, D. Garlan, and D. Siewiorek. Project Aura: Toward Distraction-free Pervasive Computing. IEEE Pervasive Computing, 1(2). 2002.

[29] R. Cerqueira, A. Ranganathan, R. H. Campbell, M. Román, C. K. Hess, and K. Nahrstedt. Gaia: A Middleware Infrastructure to Enable Active Spaces. IEEE Pervasive Computing, 1(4). 2002.

[30] X. Fu, W. Shi, A. Akkerman and V. Karamcheti. CANS: A Composable, Adaptive Network Services Infrastructure. In Proceedings of the Usenix Symposium on Internet Technologies and Systems. 2001.

[31] V. Issarny, D. Sacchetti, F. Tartanoglu, F. Sailhan, R. Chibout, N. Levy, and A. Talamona. Developing Ambient Intelligence Systems: A Solution based on Web Services. Journal of Automated Software Engineering. Vol 12. 2005.

[32] JAIN. http://java.sun.com/products/jain/.

[33] 3GPP. http://www.3gpp.org/.

[34] PARLAY. http://www.parlay.org/.

[35] Ed. T. Zahariadis and B. Doshi. IEEE Wireless Communications Magazine. Special Issue on Applications and Services for the B3G/4G Era. October 2004.

[36] J2ME. http://java.sun.com/j2me/

[37] OSGi. http://www.osgi.org/.

[38] MOBIUS: http://mobius.inria.fr/

[39] H. Muccini, A. Bertolino, and P. Inverardi. Using Software Architecture for Code Testing. IEEE Transactions on Software Engineering, 30(3). 2004.

[40] V. Issarny and A. Zarras. Software architecture and Dependability. in Formal Methods for Software Architecture, M. Bernardo, P. Inverardi (Eds.), LNCS 2804, Springer-Verlag. 2003.

[41] S. Balsamo, A. Di Marco, P. Inverardi, and M. Simeoni. Model-based Performance Prediction. Software Development: A Survey IEEE Transaction on Software Engineering. 2004.

[42] A. Bertolino and A. Polini. The Audition Framework for Testing Web Services Interoperability". In Proceedings of 31th Euromicro Conference on Software Engineering and advanced Applications, EUROMICRO 2005. 2005.