PLASTIC: Providing lightweight & adaptable service technology for pervasive information & communication

Antonia Bertolino, Wolfgang Emmerich, Paola Inverardi, Valérie Issarny, Fotis K. Liotopoulos, Pierre Plaza

PLASTIC: Providing Lightweight & Adaptable Service Technology for Pervasive Information & Communication

Antonia Bertolino[(1)], Wolfgang Emmerich[(2)], Paola Inverardi[(3)], Valérie Issarny[(4)], Fotis Liotopoulos[(5)], Pierre Plaza[(6)]

[(1)]CNR-ISTI, Pisa, Italy [(2)]UCL, London, UK [(3)]University of L'Aquila, Italy
[(4)]INRIA, Paris-Rocquencourt, France [(5)]VTRIP, Greece [(6)]Telefonica I+D, Spain

bertolino@isti.cnr.it, w.emmerich@cs.ucl.ac.uk, inverard@di.univaq.it, valerie.issarny@inria.fr, liotop@vtrip.net, pierre@tid.es

Abstract

The PLASTIC project adopts and revisits service-oriented computing for Beyond 3rd Generation (B3G) networks, in particular aiming at assisting the development of services targeted at mobile devices. Specifically, PLASTIC introduces the PLASTIC platform to enable robust distributed lightweight services in B3G networking environments through:

- *A development environment for the thorough development of SLA- and resource-aware services, which may be deployed on the various networked nodes, including handheld devices;*
- *A service-oriented middleware leveraging multi-radio devices and multi-network environments for applications and services deployed on mobile devices, further enabling context-aware and secure discovery and access to such services;*
- *A validation framework enabling off-line and on-line validation of networked services regarding functional and non-functional properties.*

1. Introduction

The PLASTIC project (http://www.ist-plastic.org) introduces the PLASTIC platform to enable robust distributed lightweight services in B3G networking environments (i.e., environments aggregating the various networking technologies now available) through both design-time and run-time development support.

Regarding design-time development support for software services, currently available solutions provide an adequate support for guaranteeing that the functional requirements are satisfied. Within PLASTIC, we propose a novel framework for the additional management of Quality of Service (QoS) properties starting at development time. In particular, the PLASTIC development environment comprises modeling tools for the functional and the non-functional behavior of services and for Service Level Agreements (SLAs), non-functional analysis tools (such as the Queuing Network Solver to obtain performance analysis from the service model), and code generation tools that take into account the Service Level Agreements and the non-functional requirements to generate WSDL specifications and Java code. A further point of novelty is the uniform management of the whole development cycle using the paradigms of Model-Driven Architecture (MDA) and UML.

Regarding run-time support for service oriented computing in B3G networks, relevant middleware-layer solutions are mostly infrastructure-centric (e.g., IMS, UMA); services are deployed in the core network and end-user terminals act as service clients. Further, effective B3G networking with vertical handover between heterogeneous networks being performed to enhance connectivity, is mostly dealt with in the network layer, possibly with the support of some network infrastructure. Within PLASTIC, we concentrate on infrastructure-less, lightweight B3G networking, in particular enabling (i) end-user mobile terminals to act as both service clients and providers, and further (ii) handover at the middleware layer on both the client and server sides. In addition, PLASTIC brings innovative communication protocols customized for the B3G networking environment, specifically introducing (i) B3G SOAP for the effective routing of SOAP messages over multi-radio, multi-network links and (ii) B3G CBR for content-based routing in large scale networks. Last but not least, the PLASTIC middleware introduces key services for service-oriented applications to be executed over B3G networks, i.e., service discovery, context awareness, and security.

Complementary to the above, PLASTIC has developed a framework for service validation, conceived for usage at both design-time and run-time. The framework includes a coherent set of tools automating validation tasks for both off-line and on-

line stages, and addressing both functional and QoS concerns. Verification and validation of Service-Oriented Architecture (SOA) is a very active research topic, and several proposals can today be found in the literature addressing service analysis and validation. However, such proposals are generally conceived for a specific objective. In PLASTIC, a concerted effort for validation has been devoted providing the opportunity to develop a consistent matrix methodology, and related toolset, which is unique in terms of comprehensiveness and flexibility.

Key elements of the PLASTIC platform are further detailed in the next 3 sections, which respectively address the PLASTIC development environment (§2), middleware (§3) and validation framework (§4). We then sketch ongoing evaluation of the PLASTIC platform through the development of e-services from various application domains (§5). We then conclude with a summary of PLASTIC contributions (§6).

2. Rigorous Service Development

In PLASTIC, we envisage that mobile application development will be broadly based on industry standard software development methods and tools, most notably UML, a Java Development Kit and the Eclipse platform. We then specifically address concerns for the development of B3G applications that are not yet well supported in current development processes and tools, most notably: (i) the handling of QoS definitions in precise service level agreements, (ii) the reasoning about QoS constraints using appropriate models, both at design and at run-time, and (iii) the development of components that are aware of the usually limited resources (most notably battery life, memory and CPU speed) on typical mobile devices.

In more detail, the PLASTIC development process reported in Figure 1, considers Service Level Specification (SLS) as part of a Service Model, as opposed to existing approaches where SLS consists, at best, in additional annotations reported on a (service) functional model. Therefore a language for precise service level specifications and service level agreements in service-oriented B3G applications has been defined [1].

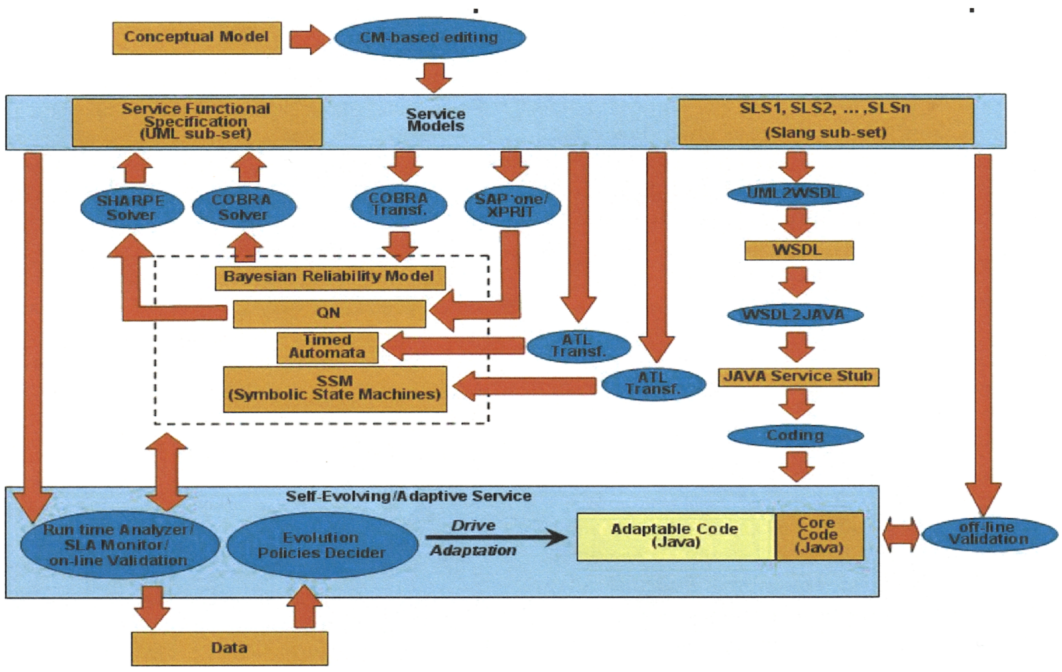

Figure 1: The PLASTIC Development Process Model

Moreover, central to the development process is the notion of *models*. *Models* are abstract views of systems, suitable for reasoning, developing, and validating a real system. Models can be functional and non-functional and can represent different levels of abstractions of the real system, from requirements to code. *Model to model transformations* permit the shift among various levels of abstractions and across several domains, possibly in an automatic way. Models analyses and transformations play an important role in PLASTIC service development. Indeed, as illustrated in

Figure 1, in PLASTIC, all the development process activities will originate from the conceptual model and exploit as much as possible model-to-model and model-to-code transformations. By taking into account the conceptual model, the service model is specified in terms of its functional specification and its SLS. The former describes behavioral aspects of the modeled service, the latter its QoS characteristics. Model-to-model transformation is performed in order to derive models for different kinds of analysis. Some models, e.g., stochastic and behavioral models, are used at development-time to refine/validate the service model (see the loop "Service Model -> Model-To-Model Trans. -> Analysis Models -> Model Solving -> Service Model" shown in the figure). Some models will also be made available at deployment- and run-time to allow the adaptation of the service to the execution context and service online validation, respectively.

Model-To-Code transformation (right-hand side of

Figure 1) is used to build both the *core* and the *adaptive* code of the service. The core code is the frozen portion of the developed self-evolving/adaptive service. The adaptive one is a *"generic code"*. A generic code embodies a certain degree of variability that makes the code capable to evolve. This code portion is evolving in the sense that, based on contextual information and possible changes of the user needs, the variability can be solved hence leading to a set of *alternatives*. A particular alternative might be suitable for a particular execution context and specified user needs, and can be selected by exploiting the analysis models available at run-time. When either a change in the context is perceived or a change of the user needs occurs, the run-time analysis is performed (on the available models) and, based on the analysis results, a new set of alternatives is synthesized and a new alternative is selected.

According to the above, the PLASTIC development environment decomposes into five main blocks [2]:

- **Modeling tools**: These are the *PLASTIC service model editor* and the *PLASTIC SLA editor*. The former allows the developer to model the functional and non-functional behavior of the service. The latter is for defining the service level specification by taking into account the feedbacks from the non-functional analysis tools.
- **Non-functional analysis tools**: These are *L-to-QN* and *QN-Solver*. The former derives, from the service model, a performance model based on Queueing Networks (QN). The latter is a QN solver for performing performance analysis.
- **Code generation tools**: *UML to HUTN* and *UML to WSDL* are used to derive, from the service model, information needed for Service Level Agreement (SLA) management and the service WSDL specification. *SLA monitor generator* and *WS-Agreement generator* generate monitors in order to check SLA at run-time. *PLASTIC model to Adaptable Java* generates skeleton of the service's code. This code is resource-aware in the sense that it adapts to possible different hosting devices.
- **Resource-aware programming tools**: These rely on a *resource-aware programming model*, which is a resource-aware extension of Java used to write the resource-aware code of the service. Then, the *Abstract Resource Analyzer* and *Customizer* are used at deployment-time in order to "extract", from the resource-aware implementation of the service, the alternative that best fits the resource constraints of the device over which the service will execute.

The PLASTIC development environment relies on the Eclipse framework, i.e., all the tools of the PLASTIC development environment are implemented as Eclipse plug-ins, and are available under open source licenses at the PLASTIC Web site. However, each tool has been developed in a modular way hence providing also an API that would allow a developer to use the tool also outside Eclipse.

3. Service Oriented Middleware

In order to aid in the effective exploitation of the rich wireless networking environment, the PLASTIC middleware empowers the service-oriented architecture with B3G networking capabilities [3]. Particularly, there are 2 main objectives for the PLASTIC middleware:

- To offer a proof-of-concept, lightweight service-oriented middleware, supporting the PLASTIC development process, for the deployment and dynamic composition of mobile, adaptable services in B3G networks. The middleware in

particular allows the deployment of PLASTIC services over a large diversity of terminals, including wireless, resource-constrained ones.

- To support advanced middleware-level functionalities for mobile adaptable services, i.e.: context-aware service management that adapts service provisioning to both resource availability and user profile, trust and security management, SLA enforcement, and information dissemination.

To achieve the above, the PLASTIC *service-oriented middleware* is specifically structured around 3 layers [4]:

- **Multi-radio middleware**: The lower layer deals with service provisioning in the B3G network, focusing on enabling the execution of services on the end-users' wireless handheld devices embedding various radio network interfaces. The layer offers the abstraction of an integrated multi-radio network through the *multi-radio networking* functionality, which comprehensively composes the various networks in reach via the embedded radio interfaces (e.g., GPRS, WiFi, Bluetooth), as identified by the *multi-radio device management* functionality. Basically, the multi-radio networking layer offers point-to-point and multicast messaging over the multi-radio network, and chooses the underlying network over which messages are to be actually sent according to application-layer requirements in terms of both functional and non-functional properties.
- **Communication middleware:** The mid layer introduces advanced communication protocols, in particular customized for the underlying multi-radio networking. *B3G SOAP* enriches traditional functionalities of a SOAP engine to allow for SOAP-based interactions in the B3G network, which includes both: (i) enabling access to services that may be in distinct networks thanks to multi-network routing and (ii) dealing with seamless mobility as long as the respective hosts of the given service's client and provider remain in reach via at least one link of the multi-radio network. In addition to B3G SOAP, *B3G content-based routing* is especially suited for open B3G networks, enabling the routing of messages according to their content and interest for it, as opposed to the addressing of specific nodes.
- **PLASTIC middleware services:** The top layer embeds advanced middleware services for the B3G network, enabling *service discovery*, *context awareness*, *security and trust*, and *publish-subscribe* in the target network

The PLASTIC middleware prototype has been developed as a set of cooperating modules, which can all be used in conjunction but also in isolation. Then, the specific middleware modules deployment within a node depends on the node's capabilities (e.g., multiple radio interfaces, resource constraints) and the requirements of the hosted service clients and providers. Further, upper middleware services are made available as Web services in the network.

Initial implementation of the PLASTIC middleware prototype is available under open source licenses from the PLASTIC Web site and is further being experimented with, through the development of e-services, as discussed in Section 5.

4. Service Validation Methodology & Tools

To enforce dependability of developed services, the PLASTIC project has undergone the development of a supporting validation methodology and associated toolset [5]. The objectives for the PLASTIC validation framework are to develop:

- A testing methodology for mobile, adaptable component-based services.
- Methodologies, both analytical and empirical, for assessing QoS of mobile, adaptable component-based services.
- A test framework to be incorporated into the PLASTIC platform.

To address the above objectives, we have articulated the PLASTIC validation strategy into two main stages, referred to as *off-line* and *on-line* testing. This two-staged strategy is in agreement with the current trend in the literature of SOA validation and accounts for two paired requirements. On one side, it is desirable for service developers to anticipate as early as possible the detection of potential problems. Therefore, off-line testing techniques exercise the software before it is released, by reproducing/faking its execution on a derived test suite. On the other side, given the complexity of the context in which B3G applications will execute, it is impossible to sample a priori (i.e., off-line) all possible ways and contexts in which a service will be invoked. This is not only because of the great variety and scale of clients foreseeable, but also due to the inherent dynamicity, mobility and heterogeneity, which characterize the network infrastructure of PLASTIC applications. Therefore, beyond the off-line testing stage carried on by the service developer, it becomes indispensable that the provider of a service continues to test the service behavior during live usage, in particular for what concerns its adaptability to context changes due to

mobility. In the PLASTIC vision, on-line testing becomes especially important as the levels of quality to be guaranteed is agreed in an SLA specification and it becomes mandatory to check that the agreement is not violated. The objective we pursue is then the development of a validation strategy for PLASTIC services, including both off-line and on-line approaches, covering both functional and non-functional properties.

Given the many types of interoperating services that are involved, the general approach that we have taken towards the above objectives has been that of not committing the validation framework to one fixed test methodology, but rather to conceive it as an open environment, which can include different methods and tools. In other words, to develop the validation framework, we deemed it more useful and farsighted to design it as an open flexible validation process, which spans over the development, deployment, and provision of PLASTIC services, rather than in the restricted sense of the instantiation of a specific testing environment. This has been extensively described in [5], which provides a detailed view on the proposed PLASTIC validation framework, with state-of-the-art overview and justification for the adopted techniques. This has further led to implement the envisaged framework into a working validation platform [6].

In more detail, during the off-line stage, services are tested in a simulated environment that reproduces functional and/or non-functional run-time conditions. The tools made available for this purpose within the PLASTIC validation framework are:

- **Jambition**: a *model-based testing tool* that automatically derives and executes invocation sequences on a service, checking whether the responses conform to a given specification, expressed as a Service State Machine. The commercial UML editor MagicDraw can be used to model a State Machine, which is then transformed into a format suitable for Jambition by the Minerva library.
- **PUPPET**: a tool to *generate stubs* implementing external services invoked by the service under development. The mock services automatically generated by Puppet exhibit a correct behavior with respect to given non-functional properties.
- **WEEVIL**: a *synthetic-workload generator* coupled with an environment for *managing the deployment and execution of experiments*. Weevil is intended to facilitate experimentation activities for distributed systems by providing engineers with a flexible, configurable, automated and, thus, repeatable process for evaluating their software on a networked testbed.

On-line validation further foresees testing of a service after deployment and during live usage. PLASTIC provides tools developed to support this phase in verifying both functional and non-functional properties:

- **Dynamo-AOP:** a framework for *monitoring functional properties* of the external services with which a BPEL process interacts to realize a composite service.
- **SLAngMon:** a light-weight technology that can dynamically detect violations of non-functional properties specified in SLAng, by means of automatically generated on-line monitors. Events related to the non-functional characteristics are logged and can be used by end-users to resolve controversies concerning SLA violations.

All the tools are available under open source licenses and can be downloaded from the PLASTIC Web site; they are now being experimented, as sketched below.

5. PLASTIC Assessment

In order to assess the added value of the PLASTIC platform, we have developed mobile e-services from various, relevant application domains, based on existing e-services, while still targeting innovative mobile application services. We have specifically been re-engineering existing applications (originally targeted for "wired" networks) as well as developing brand new software applications, all targeted to B3G mobile devices and networks (see Table 1) [7].

6. Conclusion

The PLASTIC platform ultimately integrates software development methods and tools, and middleware, enabling service provision in the open wireless environment, which decomposes into:

- Support for the development of software 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.
- A service-oriented middleware enabling infrastructure-less, lightweight B3G networking, in particular allowing (i) end-user mobile terminals to act as both service clients and providers, and further (ii) network handover at the middleware layer on both the client and server

sides. In addition, the PLASTIC middleware brings innovative communication protocols together with key enabling services, which are customized for the B3G networking environment.

- Support for off-line and on-line validation of service properties.

Scientific and technological developments that are integrated in the PLASTIC platform specifically lead to innovative results in the following areas:

- Development and provisioning of robust adaptive services for the open wireless environment,
- Service-oriented middleware for lightweight service provisioning in Beyond 3G networks,
- Testing methods and tools to validate the dependability of mobile, adaptive services.

The PLASTIC platform is now being assessed in terms of its effectiveness in promoting innovative mobile e-services, and decreasing development and usage costs. Assessment is based on the development of mobile e-services in the areas of: e-business focused on mobile CRM (Customer Relationship Management), e-government focused on a mobile e-voting service, e-health related to tele-medicine, and e-learning.

Application	Features	Added Value
e-Health	Tele-assistance applications that provide support for the provision of mobility for the users (patients) and for the assistants (i.e., doctors).	The mobility of the patients and the doctors will increase the user acceptance and application possibilities in new scenarios that are in accordance with the "mobile world" we are in.
m-Health	An application that enables mobile patients to trigger medical alarms that reach health professionals according to their specificity and availability. The application enables then the selected health professional to execute a remote diagnosis, ensuring service continuity even in cases of connection problems or changes.	Mobility is provided to both patients and health professional, increasing medical services availability, diversity and response time.
e-Business	Customer Relationship Management application that provides mobility for Technical Service Engineers (TSE) working in the automotive field. TSEs are experts paid by car manufacturers to travel from dealership to dealership to solve issues the dealer cannot (or does not want to) solve.	The application advances the state of the art integrating, among others, the following elements: SOA for B3G users, collaborative activities, data exchange across heterogeneous networks.
e-Voting	An application to facilitate a decision-making process (through e-voting) in a secure and fault-tolerant manner, as part of a Crisis Management System. A small number of users form an ad hoc network and implement a distributed e-voting protocol with their mobile handheld devices.	Secure implementation of a mutual authentication protocol in an ad hoc mobile network. A trusted critical decision-making process, implemented over B3G networks.

Table 1: Developing e-services using the PLASTIC platform

Acknowledgments: This work is part of the IST PLASTIC project (http://www.ist-plastic.org/), funded by the European Commission under the FP6 contract number 026955.

References: The references below address the design and prototype implementation of the PLASTIC platform elements, together with assessment of the proposed contributions with respect to state of the art. Those references and related publications may be found from the PLASTIC Web site at http://www.ist-plastic.org/.

[1] PLASTIC consortium. Deliverable D2.1. SLA language and analysis techniques for adaptable and resource-aware components.

[2] PLASTIC consortium. Deliverable D2.3. Development process and tools: assessment and revision.

[3] PLASTIC consortium. Deliverable D3.1. Middleware Specification and Architecture.

[4] PLASTIC consortium. Deliverable D3.3. Middleware: Assessment and Revision.

[5] PLASTIC consortium. Deliverable D4.1. Test Framework Specification and Architecture.

[6] PLASTIC consortium. Deliverable D4.3. Test framework: Assessment and Revision.

[7] PLASTIC consortium. Deliverable D5.1. Integrated PLASTIC platform and supporting user guide and training courses.