Towards Application Suitability for PvC Environments*

Andres Flores[1] and Macario Polo[2]

[1] Departamento de Ciencias de la Computación, Universidad Nacional del Comahue, Buenos Aires 1400, 3400 Neuquén, Argentina
[2] Escuela Superior de Informática, Universidad de Castilla-La Mancha, Paseo de la Universidad 4, Ciudad Real, España

Abstract. Pervasive Computing Environments should support the feeling of continuity on user's daily tasks. This concept relies on the availability of different resources (in this work that will mean availability of applications). We propose a framework for a component-based integration process, centred on the concept of composing/adapting applications at run-time. Central to the problem is the task of component assessment, at syntactic and semantic levels. Our aim is to apply metadata-based techniques and a process-oriented simulation which resorts to well establlished verification tools.

1 Introduction

Component-based Software (CBS) refers to the possibility of transparently integrate off-the-shelf (OTS) components on a new target environment in order to satisfy a given functionality. Then reusability is one of the main aspects to be supported since components are used "as they are found" instead of being modified [1]. Hence, the need for reliability on components functionality and components inter-connection. Important proposals have been delivered for CBS verification and validation [2, 3, 1], though much work have to be accomplished yet in this area.

Pervasive Computing Environments (PvCEnv's) require special considerations on reliability. As pervasive computing implies computation becoming part of the environment, specific and disparate software as well as a variety of heterogeneous computing devices have to be interrelated to allow access to information from anywhere and at anytime in a secure manner [4].

The user's relationship to computation changes from the traditional tasks performed "on the computer". Instead, individuals may come to expect certain services facilitated by the environment, which will aim at providing a feeling of "continuity" in their daily tasks [5]. Such users do not expect to find that the surrounding place is in fact a constrained environment [6]. Thus applications should not be set to a fixed collection. Here we explore the possibility to compose applications 'on demand' and maintain their suitability upon different changes. This could be achieved by a run-time assembly of applications from disparate OTS components [7], which allow to make adjustments by

* This work is supported by the CyTED project VII-J-RITOS2, the UNCo MPDSbC project 04-E059 and the UCLM MAS project TIC 2003-02737-C02-02.

replacing components to satisfy the computational demands of the user as s/he moves in a place.

Components interconnection motivates an inter-operability pattern involving the consideration of different levels of information about the components [8] – e.g. syntactic aspects are undertaken in [9], whereas in [10] the focus is on the semantic level. Our emphasis is at the semantic level but syntactic aspects are considered as well. Our approach relates to the concept of *Semantic Interoperability* and our assumption is that to deal properly with these semantic aspects it is important the consideration of a formal framework to reason about important aspects of the Component-based Integration Process. Our work aims to develop a solution to address the problems faced in the integration process which consider specific phases like: '*qualification*', '*adaptation*', '*assembly*' and '*integration*'. Currently we are focused on the Qualification phase, for which different techniques have been applied in connection to the development of a Component Assessment procedure.

On a PvCEnv the usual scenario implies users changing from one operational context to another. This may involve to use a different device and expect to continue working under the same or similar application. Hence, requirements are invariably updated as well as those resources that need to be accessible. When a required application is not available or degrades its suitability, the '*assembly*' could be initiated from a previously executed selection procedure of proper components. They could be fetched from a repository or being discovered from a mobile device. In any case evaluation should be rigorous [7].

2 Component Assessment

Our Assessment Procedure intends to compare behavioural aspects from components against a given set of requirements. The requirement specification is assumed in the form of a component interface – the necessary set of component services. Thus our approach, which compares components interfaces, actually evaluates a component against an expected set of services.

At a syntactic level we evaluate matching on the signature of a service – e.g identifiers, parameters, and data types. A preliminary approach was developed on a previous work [7], which is currently being extended. Additionally, components will be enriched by adding meta-data – an adaptation mechanism called *instrumentation* [11]. Meta-data has been used in several approaches as a technique to make testing easier [1, 12, 13]. Currently we are focusing more on the semantic level of interoperability.

Our first concern is to add '*assertions*' to abstract out the black box functionality hidden on components. Also, the '*usage protocol*' that describes the expected order of invocations for component services, will be included in the form of regular expressions. This technique has been applied on inter-class testing [14], and also on descriptions for components [15, 10]. We expect to succesfully adapt the idea to our framework.

Suppose, for example, post-conditions on services from two similar components. They should relate to a similar structure and semantic. Hence, they could be thought as being one a '*clone*' of the other. Thus we apply some algorithms based on Abstract Syntax Trees (AST) from [16], which were originally intended to detect similar pieces of code (clones) on existing programs. Then compatibility for assertions and the usage protocol is carried out by generating ASTs.

It is also our intention to incorporate a temporal dimension to our approach. As some authors have pointed out [17, 18], temporal aspects could also be helpful to achieve a more accurate integration process. A component can be characterized by the states it goes through during operation. We plan to explore whether all possible states are achievable on a component under evaluation with respect to their counterparts as in the requirement specification. We are analysing current languages and tools in order to develop this aspects of the system.

We have started experimentation on such an approach using a process-oriented description by using Promela language associated to the SPIN verification tool. Thus components may include a Promela-based specification of their behaviour which can be used to explore the components possible states on the SPIN tool. Statements about the expected behaviour can be writte and checked using the language of Propositional Linear Temporal Logic (PLTL) [19] based query. On a previous work [7] we have applied PLTL to verify the dynamic aspects of components behaviour.

3 Implementation Alternatives

Well-known platforms as Microsoft .NET and Enterprise Java Beans provide mechanisms to enable meta-data incorporation into components. Information about components - its structure and the state of their corresponding instances - can be recovered at runtime by using *Reflection* on both, Microsoft .NET and Enterprise Java Beans. Particularly .NET includes the possibility of adding *Attributes*, which is a special class intended to provide additional information about some design element as a class, a module, a method, a field, and so on.

In order to put our ideas to test and assess the viability of our approach, we have recently developed a preliminary prototype on .Net. Some functions from the framework process were implemented, like the Recovery procedure from a Component Repository. Also a first draft of what could be the Adaptation or Tailoring techniques to be applied on components were represented. The Assessment procedure was also partially implemented by considering the interface, assertions and usage protocol matching.

4 Conclusions

The main aim behind this project is to automate a Component-based Integration Process for PvCEnv's. We have presented a phased scheme and explained the Component Assessment procedure related to the Qualification phase. For this we apply metadata-based techniques in order to address Interoperability at a Semantic level.

We have also discussed some alternatives in order to implement different functions from the framework process, by the use of well-known platforms like .Net and EJB. The experience gained from building a simple prototype on .Net has been rewarding but more experimentation is necessary in order to recognize not only the efficiency level but mainly effectiveness on supporting reliability.

Our work is at a preliminary stage and currently we are exploring different techniques to improve our process on efficacy and reliability. As reliability is our main concern, selecting appropriate methods, techniques and languages, must be accurately accomplished. Our next step is related to the accomplishment of the semantic level for component assessment. We believe the complementary procedure with incorporation of temporal aspects will be a significant step to achieve an adequate deployment.

Acknowledgments

We would like to thank Dr. Juan Carlos Augusto, from School of Computing and Mathematics, University of Ulster, Newtownabbey, UK. His participation provides a meaningful complementary view giving accuracy and confidence to the project.

References

1. Cechich, A., Piattini, M., Vallecillo, A.: Component-based Software Quality: Methods and Techniques. Volume 2693 of LNCS. Springer-Verlag (2003)
2. Weyuker, E.J.: Testing component-based software: A cautionary tale. IEEE Software (1998) 54–59
3. Harrold, M., Liang, D., Sinha, S.: An Approach to Analyzing and Testing Component-based Systems. In: ICSE'99, 1st Wrkshp on Testing Distr. CBS, Los Angeles, CA., US (1999)
4. Garlan, D., Schmerl, B.: Component-based Software Engineering in Pervasive Computing Environments. In: Workshop on Component-Based Software Engineering: Component Certification and System Prediction (held during 4th ICSE'01), Toronto, Canada (2001)
5. Wang, Z. and Garlan, D.: Task-Driven Computing. Technical Report CMU-CS-00-154, Carnegie Mellon University, School of Computer Science (2000) http://reports-archive.adm.cs.cmu.edu/anon/2000/abstracts/00-154.html.
6. Garlan, D.e.: Software Architecture-based Adaptation for Pervasive Systems. In: ARCS'02. Volume 2299 of LNCS., Karlsruhe, Germany (2002) 67–82
7. Flores, A., Augusto, J.C., Polo, M., Varea, M.: Towards Context-aware Testing for Semantic Interoperability on PvC Environments. In: IEEE 17th SMC'04, special session: CRIPUC, The Hague, Netherlands (2004) 1136–1141
8. Euzenat, J.: Towards a principled approach to Semantic Interoperability. In: Wrkshp on Ontologies and Inf. Sharing, (IJCAI'01), Seattle, US (2001) 19–25
9. Koné, O.: An Interoperability Testing Approach to Wireless Application Protocols. JUCS, Journal of Universal Computer Science **9** (2003) 1220–1243
10. Pahl, C.: An Ontology for Software Component Matching. In: FASE'03. LNCS 2621, pp. 6-21, Berlin, Germany, Springer-Verlag (2003)
11. Flores, A., Polo, M.: Dynamic Assembly & Integration on Component-based Systems. In: 4th JIISIC, Madrid, España (2004) 349–360
12. Cechich, A., Polo, M.: COTS Component Testing through Aspect-based Metadata. In: Building Quality into Components - Testing and Debugging. Springer-Verlag (2005)
13. Orso, A., Harrold, M., Rosenblum, D.: Component Metadata for Software Engineering Tasks. In: 2nd EDO'00, Springer-Verlag LNCS 1999, pp.129-144, Davis, CA, USA (2000)
14. Kirani, S.: Specification and Verification of Object-Oriented Programs. PhD thesis, Computer Science, University of Minnesota, Minneapolis, USA (1994)
15. Brada, P.: Towards Automated Component Compatibility Assessment. In: 6th Wrkshp on Comp-oriented Prog, at ECOOP'01, Budapest, Hungary (2001)

16. Baxter, I., Yahin, A., Moura, L., Sant'Anna, M., Bier, L.: Clone Detection Using Abstract Syntax Trees. In: ICSM'98, pp. 368-377, Maryland, USA (1998)
17. Chen, H., Finin, T., Joshi, A.: Semantic Web in the Context Broker Architecture. In: 2nd IEEE PerCom'04, Orlando, US (2004) 277
18. Ranganathan, A., Campbell, R.: An infrastructure for context-awareness based on first order logic. Personal and Ubiquitous Computing **7** (2003) 353–364
19. et.al., B.B.: Systems and Software Verification (Model Checking Techniques and Tools). Springer-Verlag (1999)