Towards a Tool-based Development Methodology for Sense/Compute/Control Applications (Poster)

Damien Cassou, Julien Bruneau, Julien Mercadal, Quentin Enard, Emilie Balland, Nicolas Loriant, Charles Consel

Towards a Tool-based Development Methodology for Sense/Compute/Control Applications

Damien Cassou Julien Bruneau Julien Mercadal Quentin Enard Emilie Balland
Nicolas Loriant Charles Consel

University of Bordeaux / INRIA / LaBRI, France
first-name.last-name@inria.fr

Abstract

This poster presents a design language and a tool suite covering the development life-cycle of a *Sense/Compute/Control (SCC) application*. This language makes it possible to define the architecture of an application, following an architectural pattern commonly used in SCC applications. Our underlying methodology assigns roles to the stakeholders, providing separation of concerns. Our tool suite includes a compiler that takes design artifacts written in our language as input. The compiler generates customized support for subsequent development stages, namely implementation and test. In doing so, it ensures the conformance between the architecture and the code. Our tool suite also includes a simulator for testing SCC applications, without requiring code modification. Our methodology has been applied to a wide spectrum of areas, such as building automation, advanced telecommunications, and health-care.

Categories and Subject Descriptors D.2.11 [*Software Engineering*]: Software Architectures—Domain-specific architectures; D.3.4 [*Software Engineering*]: Processors—Code generation

General Terms Design, Languages

Keywords Domain-Specific Languages, Architecture Description Languages, Generative Programming, Methodology

1. Introduction

Sense/Compute/Control (SCC) applications are applications that interact with a physical or computing environment. SCC applications are being deployed in a growing number of areas, including building automation, assisted living, robotics, ubiquitous computing, and autonomic computing. These systems involve a wide range of devices and software components, communicate using a variety of protocols, and rely on intricate distributed systems technologies. Besides requiring expertise on underlying technologies, developing an SCC application also involves domain-specific architectural knowledge to collect information relevant for the application, process it and perform actions. To improve the development of SCC applications, we propose a tool-based methodology that relies on the architectural description for guiding each development phase.

SPLASH'10, October 17–21, 2010, Reno/Tahoe, Nevada, USA.

2. DiaSuite methodology

Figure 1 illustrates the different stages of our development methodology and its associated tools.[1]

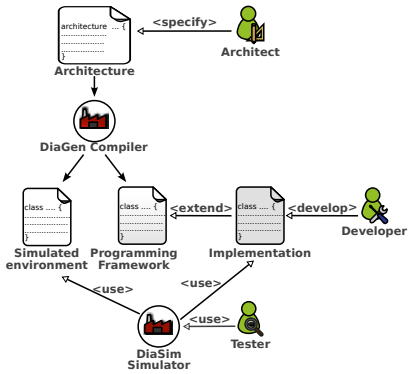

Figure 1. Development support provided by DiaSuite.

2.1 Specifying the architecture

To specify the architecture of an SCC application, we propose DiaSpec, a domain-specific Architecture Description Language (ADL). This domain-specific ADL is associated with an architectural pattern illustrated in Figure 2 that consists of three types of components: *entities* send information sensed from the environment to the context layer through data sources; *contexts* refine (aggregate and filter) the sensed

[1] http://diasuite.inria.fr

information provided by the entities; *controllers* interpret the information provided by the contexts to issue orders to the entities; finally, entities trigger actions on the environment. Our domain-specific ADL also includes declarations dedicated to error handling. These architecture-level declarations provide a separation between functional and error-handling concerns [6].

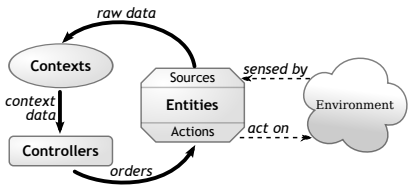

Figure 2. SCC Architectural pattern used in DiaSuite.

2.2 Implementation

We leverage the architecture description to provide dedicated support to the developers. This support takes the form of a Java programming framework, generated by the DiaGen compiler [3]. This framework guides the developer by providing high-level abstractions of low-level details (*e.g.,* the distributed systems technology) and ensures the conformance of the resulting implementation with the architectural specification. The generated programming framework also contains support for signaling, propagating and treating errors, which makes the programming of error handling more rigorous and systematic [6].

2.3 Testing

DiaGen generates a simulation support to test SCC applications before their actual deployment. An application is simulated with DiaSim [2], without requiring any code modification. DiaSim provides a graphical editor to define simulation scenarios and a 2D-renderer to monitor simulated applications. Furthermore, simulated and real entities can be mixed. This hybrid simulation enables an application to migrate incrementally to an actual environment.

3. Ongoing and future work

We have successfully applied our methodology to a variety of SCC applications in areas including advanced telecommunications [1], home/building automation [4], and healthcare [5]. Presently, this work is being expanded in various directions.

Enhancing the design phase. One direction consists of widening the scope of the DiaSpec language by introducing non-functional concerns (*e.g.,* fault-tolerance, safety and security) at the architectural level. For example, we are exploring how to enrich the error-handling mechanism with fault tolerance strategies.

Enhancing the testing phase. An ongoing work aims to simplify the testing phase by automatically generating a dedicated unit-testing framework. In accordance with the architect, a tester could then describe the desired behavior of each component separately, even before the implementation has started. Another direction concerns the simulation tool. Simulating natural phenomena like heat propagation can be quite complex as they involve mathematical equations. We are actively working on easing simulation of these phenomena by leveraging Acumen [7], a DSL for describing differential equations.

Enhancing the evolution phase. Our approach permits late changes to the architecture; a change in the architecture triggers a compile-time error in the developer's code. We believe the architect could be provided with refactoring tools that would ease changes in the developers' code.

References

[1] B. Bertran, C. Consel, W. Jouve, H. Guan, and P. Kadionik. SIP as a universal communication bus: A methodology and an experimental study. In *ICC'10: Proceedings of the 9th International Conference on Communications*, Cape Town, South Africa, 2010.

[2] J. Bruneau, W. Jouve, and C. Consel. DiaSim: A parameterized simulator for pervasive computing applications. In *Mobiquitous'09: Proceedings of the 6th International Conference on Mobile and Ubiquitous Systems: Computing, Networking and Services*, pages 1–10, Toronto Canada, 2009. IEEE Computer Society.

[3] D. Cassou, B. Bertran, N. Loriant, and C. Consel. A generative programming approach to developing pervasive computing systems. In *GPCE'09: Proceedings of the 8th International Conference on Generative Programming and Component Engineering*, pages 137–146, Denver, CO, USA, 2009. ACM.

[4] D. Cassou, J. Bruneau, and C. Consel. A tool suite to prototype pervasive computing applications (demo). In *PERCOM'10: Proceedings of the 8th International Conference on Pervasive Computing and Communications*, pages 1–3. IEEE Computer Society, 2010.

[5] Z. Drey, J. Mercadal, and C. Consel. A taxonomy-driven approach to visually prototyping pervasive computing applications. In *DSL WC'09: Proceedings of the 1st Working Conference on Domain-Specific Languages*, volume 5658, pages 78–99, 2009.

[6] J. Mercadal, Q. Enard, C. Consel, and N. Loriant. A domain-specific approach to architecturing error handling in pervasive computing. In *OOPSLA'10: Proceedings of the 25th International Conference on Object Oriented Programming Systems Languages and Applications (To appear)*, Reno, NV, USA, 2010.

[7] A. Y. Zhu, J. Inoue, M. L. Peralta, W. Taha, M. K. O'Malley, and D. Powell. Implementing haptic feedback environments from high-level descriptions. In *ICESS'09: Proceedings of the 6th International Conference on Embedded Software and Systems*, pages 482–489, Washington, DC, USA, 2009. IEEE Computer Society.