A Framework for Developing Component-based Applications with Temporal Analysis Capabilities

Francisco Sánchez-Ledesma, Juan Pastor, Diego Alonso and Francisca Rosique
Division of Systems and Electronic Engineering (DSIE), Universidad Politécnica de Cartagena, Campus Muralla del Mar, E-30202, Cartagena, Spain

Keywords: Software Engineering, Component-based Software Development, Model-driven Software Development, Framework, Software Design Patterns, Real-time, Temporal Analysis.

Abstract: Reactive system design requires the integration of structural and behavioral requirements with temporal ones (along with V&V activities) to describe the application architecture. This paper describes an implementation framework for component-based applications that provides developers with great control over application concurrency (number of threads and their characteristics), the computational load assigned to them, and allows the temporal analysis of the applications developed with the framework. The paper presents an improved version of a framework previously developed, putting it in the context of a global Model-Driven Software Development approach for developing, analyzing and generating code for reactive applications.

1 INTRODUCTION AND PREVIOUS WORK

The best way to deal with complexity in software development is to raise the abstraction level of modeling elements, built systems from independent modules that interact with each other only through their interfaces. There is no single definition of what is in fact a module and each programming paradigm offers different answers, considering different levels of abstraction and granularity.The approach chosen in this work is the Component Based Software Development (CBSD), in which a software component is a composition unit with well-defined interfaces and a explicit use context.

The work presented here provides an approach that uses techniques of component based development to reduce the complexity of applications, not forgetting the real-time requirements and focusing on the temporal analysis.There are other initiatives similar to the described here, which also consider and discuss together the issues of CBSD and real-time, as (Barros et al., 2011) and Palladio (Becker et al., 2009) in general and (Schlegel, 2008) in particular for robotics.

In the above context of software development for components based real-time systems, it was decided to follow an approach of Model-Driven Software Development (MDSD) to model this kind of application; since MDSD is a technology that provides both conceptual and technological support for modeling applications and for code generation. The MDSD approach has been used to integrate the CBSD approach with RT issues, as well as with the generation of both executable code and analysis models. By using MDSD have been developed in Eclipse (open source development platform) the tools necessary to support the development process.

Although this article focuses on the design of a framework, called MinFr, and the analysis of the temporal characteristics of the component application; a brief description of the overall approach is, however, necessary to understand the rest of the paper. Our global development approach starts by modeling the architecture of the application using the CBSD approach, and then use a series of model transformations to generate both analysis models and executable code.

Though any modeling language can be used for performing the first step, we developed our own modeling language, entitled V^3CMM(Alonso et al., 2008). The V^3CMM language provides three complementary but loosely coupled views that allows designers to define and connect software components, namely: (1) an *architectural view* to define components (interfaces, ports, services offered and required, composite components, etc.); (2) a *coordination view* to specify component behavior, based on timed automata theory (Baier and Katoen, 2008); and finally

Sánchez-Ledesma F., Pastor J., Alonso D. and Rosique F..
A Framework for Developing Component-based Applications with Temporal Analysis Capabilities.
DOI: 10.5220/0004023603150318
In *Proceedings of the 7th International Conference on Software Paradigm Trends* (ICSOFT-2012), pages 315-318
ISBN: 978-989-8565-19-8

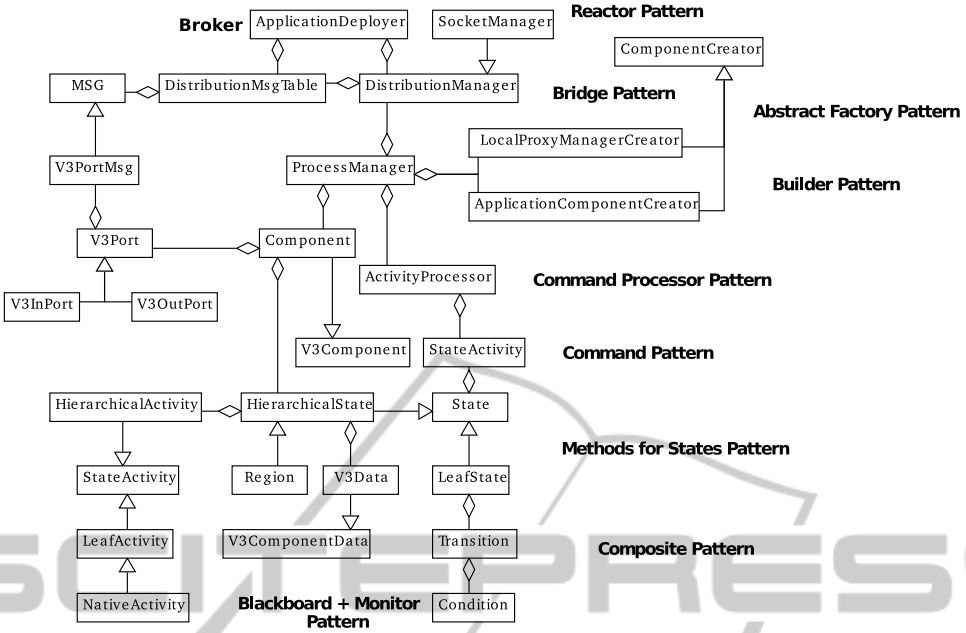

Figure 1: Simplified class diagram of the developed framework showing some of the patterns involved in its design.

(3) the activity code is associated to the state. A detailed description of the main characteristics of this language and an explanation of the reasons that led us to its development are not in the scope of this paper, but they can be found in (Iborra et al., 2009).

In order to ease the generation of executable code from the V^{3}CMM models, an OO framework was designed and implemented. Such framework provides an OO interpretation of the CBSD concepts that allows translating the component based (CB) designs into OO applications. A previous implementation of MinFr with distribution support was described in (Sanchez-Ledesma et al., 2011). This paper presents an improved version of the cited work with temporal analysis capabilities.

The rest of the paper is organized as follows: section 2 describe the way in which CBSD modeling elements (components, ports and timed automata) have been translated into OO concepts to generate executable programs. Section 3 describes how the analysis has been addressed temporal component applications using, in this case, the tool Cheddar (Singhoff et al., 2009). Finally, Section 4 presents conclusions and future work.

2 MinFr DESIGN

The design and documentation of MinFr was carried out using design patterns, which is a common practice in Software Engineering (Buschmann et al., 2007b). A pattern sequence has been followed in order to meet the following architectural drivers:

(AD1) Control over *concurrency policy*: thread number, thread spawning (static vs. dynamic policies), scheduling policy (fixed priority schedulers vs. dynamic priority scheduler), etc. (AD2) Control over the *allocation of activities to threads*, that is, control over the computational load assigned to each thread, since V^{3}CMM considers the activity associated to a state as the minimum computational unit. MinFr allows allocating all the activities to a single thread, allocating every activity to a different thread, or a combination of both policies. (AD3) Facilitate the instantiation of MinFr from the input CBSD model. (AD4) Control over the communication mechanisms between components (synchronous or asynchronous). (AD5) Control over the component distribution and deployment.

Among the aforementioned drivers, the main one is the ability to define any number of threads and control their computational load. This computational load is mainly determined by the activities associated to the states of the timed automata. In order to achieve this goal, the COMMAND PROCESSOR architectural pattern (Buschmann et al., 2007a) and its highly coupled COMMAND pattern (Gamma et al., 1995) have been selected, and they were the firsts to be applied in MinFr design. The COMMAND PROCESSOR pattern separates service requests from their execution, by defining a thread (the command processor) where the requests are managed as indepen-

dent objects (the commands). These patterns provide the required level of flexibility, since they impose no constraints over command subscription to threads, number of commands, concurrency scheme, etc. The roles defined by these two patterns are realized by the classes **`Activity_Processor`** and **`State_Activity`**, respectively. (see Figure 1).

Another key aspect, related to AD4, is to provide an OO implementation of the timed automata compatible with the selected patterns for concurrency control, in order to integrate it in the scheme defined by the aforementioned COMMAND PROCESSOR pattern. It is also an aspect that has a great influence on the whole design, since timed automata model the behavior of the components. We decided that both regions and states should be treated homogeneously, and thus we selected a simplified versions of the COMPOSITE pattern. The timed automata is managed following the METHODS FOR STATES pattern (Buschmann et al., 2007a), and the instances of the classes representing it are stored in a hash table. The roles defined by this pattern are realized by the classes **`State`**, **`Hierarchical_State`**, **`Region`** and **`Leaf_State`**.

Each activity associated to a state of the timed automata is implemented as an object, following again the COMMAND pattern, represented by the class **`State_Activity`**. In this way, activities can be allocated to any command processor. This constitutes the link between concurrency control and timed automata implementation, since activities play roles in both patterns. The distinction between states and regions led us to define two hierarchies of **`State_Activity`**, which were implemented following the STRATEGY pattern: those associated to leaf states (represented by the root class **`Leaf_Activity`**), and those activities associated to regions (represented by the class **`Hierarchical_Activity`**). The latter is aimed at managing the region states and transitions, and thus is provided as part of MinFr. The formers, are related to (i) the activities defined in the V^{3}CMM models, represented by **`Native_Activity`** subclasses, and (ii) activities to manage the flow of data and control among component through their ports.

3 TEMPORAL BEHAVIOR ANALYSIS

From MinFr instantiation model, obtained from the V^{3}CMM components application model, we proceed to the generation of a third model, called the deployment model. With the deployment and MinFr model can be generated a Cheddar analysis model and the application code. The transformation that generates the application code creates instances of framework's classes that are necessary for building components and generates the skeleton of the activities that must be completed by the programmer. The deployment model describes how it will distribute the source application into two parts: compute nodes and concurrent processes.

Once set up and completed the model of application deployment, we proceed to generate the Cheddar analysis model. Due to the XML format used by the analysis tool, it has been developed a model to text transformation using Xtend to generate the file with the information necessary to carry out the analysis. The deployment model explicitly defines the threads (Threads), which correspond directly to activities in the Cheddar model(Tasks). The model to text transformation estimates the period of each one of these threads as the greatest common divisor of all the periods of all region activities assigned to the thread. Each MinFr thread behaves as an cyclic executive and, therefore, it is possible to assign regions with different periods to the same thread.

Shared resources are not explicitly defined in the deployment model, but it is possible to determine which are potentially shared resources. The candidates for shared resources between threads are: events, ports and port connections. Events can be accessed by the activities and transitions, ports can be accessed by the activities, transitions, and the update region activity of each component and connection ports can be accessed by the update region of each component. To determine whether a resource is shared by more than one thread, it is necessary to analyze MinFr model to determine what resources each thread accesses.

The transformation from the V^{3}CMM model to MinFr model adds a number of extra items that are necessary to define some details of the application execution. It should be noted the additional region that is added to each component and is responsible for managing the ports and communication between components. Compliance with the requirements of MinFr, implementation of the code that manages the operation of it should be under the control of the developer. Therefore, the developer must also allocate these additional regions to the threads, like any other region of the application. In the case of the update region, determining the resources to be shared is performed in a similar manner, taking into account that in this case the update region has access to all ports and ports connections that the component contains. Thus it is possible to identify candidates for shared resources. Shared resources will only be those that are accessed from different threads.

4 CONCLUSIONS AND FUTURE WORK

This paper has described an approach to provide a run–time support (framework) to a CB approach for modeling applications with RT requirements. To do so, it has been necessary to provide an OO interpretation of the high–level architectural concepts (components, ports, timed automata, etc.), providing enough flexibility to also control application concurrency characteristics in order to take into account RT requirements.

The adoption of a pattern-driven approach has greatly facilitated the design of such framework. In addition, the selected patterns have been described like a pattern story. A further step would be the definition of a pattern sequence, which comprises and abstracts the aforementioned pattern story, so that developers can use it in other applications as long as they share similar requirements.

This paper also has described the evolution of the ongoing work, incorporating the ability to distribute components to an OO framework that support CB development. Temporal analysis capacity was added through a model transformation for generating input models for analysis tools. Having added temporal analysis capacities to MinFr, determines that the most urgent future work is to validate MinFr, using it in larger applications that comprise both safety critical requirements in their reactive behavior and intensive processing. In this way, it is crucial to integrate the algorithms libraries offered by other robotic frameworks. This integration is problematic, as is explained in (Makarenko et al., 2007), but it is crucial to make the approach (not only MinFr) attractive enough to the robotics community.

ACKNOWLEDGEMENTS

This work has been partially supported by the Spanish CICYT Project EXPLORE (ref. TIN2009-08572), the Séneca Project MISSION-SICUVA (ref. 15374/PI/10), and the Spanish MEC FPU Program (grant AP2009-5083).

REFERENCES

Alonso, D., Vicente-Chicote, C., and Barais, O. (2008). V3Studio: a Component-Based architecture modeling language. In *15th Annual IEEE International Conference and Workshop on Engineering of Computer Based Systems*, pages 346–355. ieee.

Baier, C. and Katoen, J. (2008). *Principles of Model Checking*. The MIT Press.

Barros, L., Lopez, P., and Drake, J. M. (2011). Design of real-time component-based applications on open platforms. In *37th Euromicro Conference on Software Engineering and Advanced Applications (SEAA)*, pages 65–72.

Becker, S., Koziolek, H., and Reussner, R. (2009). The palladio component model for model-driven performance prediction. *Journal of Systems and Software*, 82(1):322.

Buschmann, F., Henney, K., and C. Schmidt, D. (2007a). *Pattern-Oriented Software Architecture, Volume 4: A Pattern Language for Distributed Computing*. John Wiley and Sons Ltd.

Buschmann, F., Henney, K., and Schmidt, D. (2007b). *Pattern-Oriented Software Architecture, Volume 5: On Patterns and Pattern Languages*. John Wiley and Sons Ltd.

Gamma, E., Helm, R., Johnson, R., and Vlissides, J. (1995). *Design patterns: elements of reusable object-oriented software*. awp.

Iborra, A., Alonso, D., Ortiz, F., Franco, J., Snchez, P., and lvarez, B. (2009). Design of service robots. *IEEE*.

Makarenko, A., Brooks, A., and Kaupp, T. (2007). On the benefits of making robotic software frameworks thin. In *Proc. of the IEEE/RSJ Int. Conf. on Intelligent Robots and Systems (IROS'07)*. ieee.

Sanchez-Ledesma, F., Pastor, J., Alonso, D., and Rosique, F. (2011). An implementation framework for component-based applications with real-time constraints: extensions for achieving component distribution. In *Proc. of the 6th International Conference on Software and Data Technologies (ICSOFT 2011)*, pages 290–293. SciTePress.

Schlegel, C. (2008). The challenge of real time robotics behavior: An applied research perspective. In *Proc. of the 3rd Int. Workshop on Software Development and Integration in Robotics (SDIR'08)*.

Singhoff, F., Plantec, A., Dissaux, P., and Legrand, J. (2009). Investigating the usability of real-time scheduling theory with the cheddar project. *Journal of Real Time Systems*, 43(3):259–295.