STUDIA UNIV. BABEŞ–BOLYAI, INFORMATICA, Volume **LVI**, Number 3, 2011

BUILDING BLOCKS DEV STUDIO. A TOOL FOR COMPONENT BASED DEVELOPMENT

PAUL HORAŢIU STAN

Abstract. This paper presents a technique inspired from hardware engineering that can be applied in software engineering in order to develop flexible and modular systems based on independent components, these components can be written on different programming languages. A graphical user interface tool has been developed during this research. The *BB-DevStudio* combines theoretical results presented in [2] and [1] in order to generate the source code for independent components and to support communication between Java and .NET components. At the end of the article a demo application is presented together with advantages and constraints of the proposed solution.

1. Introduction

In hardware development process, an engineer can use many integrated circuits in order to develop a complex system, for instance he/she can use multiplexers, counters, logic AND, logic OR, memory, register etc, these components are interconnected on a main board and become a complex system. The counter or the multiplexer or other components are developed by a third party company. This principle can be applied both on hardware and software development process, for example a software system can have many independent components that should be interconnected on a main board in order to become a complex system. These components can be developed by a third party company. Nowadays there are many third party components that are used in software systems, but the connections between them are made by a developer, on the other hand a person should write a program that will use the third party components, this can introduce many errors; more than this,

Received by the editors: March 30, 2011.

2010 *Mathematics Subject Classification.* 68N15.

1998 *CR Categories and Descriptors.* D.2.11 [**Software**]: Software Engineering – *Software Architectures*; D.2.13 [**Software**]: Software Engineering – *Reusable Software*.

Key words and phrases. Component Based Development, Automatic Source Code Generation, Software Architectures.

changing the code written by a developer is an expensive process that needs time and money [1].

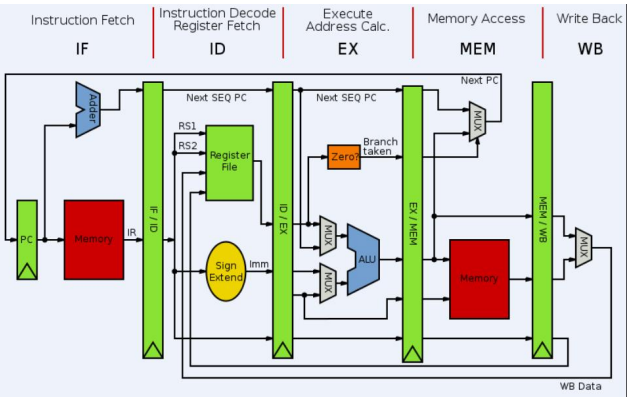

FIGURE 1. MIPS Architecture

2. The problems

The general context of this article is focused on: (1) how to access remote objects developed for different programming languages in a transparent way, like they were written for client programming language and (2) presenting a technique for automatically generate the source code for a software component that contains subcomponents.

Regarding (1) the problem is: how the parameters can be passed to the remote methods without being necessary to serialize them on client platform and restored on remote platform in order to be used there. The main idea presented in this article is that each object should be executed in the address space of the process which has created itself.

Regarding (2) the problem is: based on an XML component specification file a skeleton source code should be generated, after that a developer can add the source code that implements the business logic.

The (1) and (2) should be combined in order to support source code generation for components written on different platforms that can communicate in a transparent way.

This section is composed by three parts, first presents the current interoperability and component based approaches and the second describes the proposed solution. The proposed solution has two parts: (1) the interoperability solution and (2) the component based solution.

3. Proposed approach

3.1. **Proposed Architecture for platform interoperability.** The proposed solution is based on the Proxy design pattern presented in [16].

In the proposed solution for each remote object a proxy object is generated on the client side. When the client platform creates the proxy then the server creates the real object and store it in a hash table based on an automatically generated identifier. When the client side platform erase the proxy object, for instance the garbage collector decides to remove from memory the unreferenced objects then the server side platform will remove the real object from the remote objects hash table. When a client object invokes a method on the proxy object the request is redirected to the remote object.

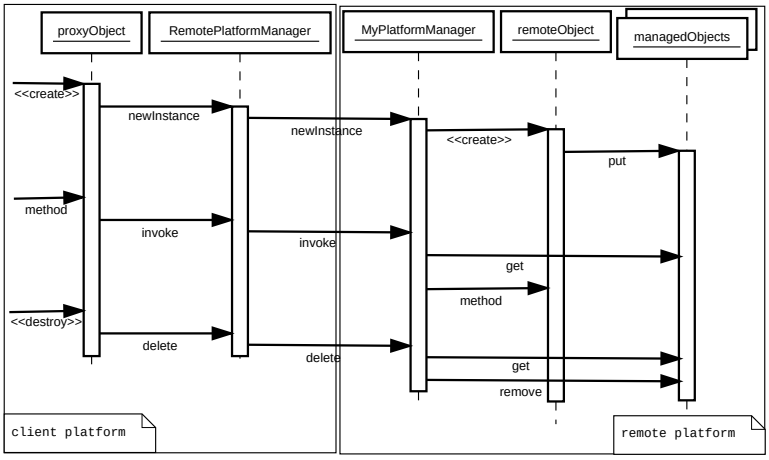

Figure 2. Communication Architecture

3.2. **Component Based Solution Architecture.** Figure 5 presents a print screen with the main architecture of a software system developed using Building Blocks framework, this tool is a result of the proposed research.

Two ports *O1* and *I1* can be connected if they have the same data type *T*. The source code for the *demoIndepComp* will be automatically generated based on a connection specification file, which is an XML file.

An independent component is an instance of a class. The component class should implement a standard interface called *IndependentComponent*. The component communicates with the environment using properties and events.

Figure 3 presents the conceptual model of the proposed solution. A component can have subcomponents, properties, links and subcomponent links. The direction of a property can be: (1) input, (2) output and (3) input/output.

A link is a connection between two properties of two subcomponents and a subcomponent link is a connection between a property of a subcomponent and a property of the parent component.

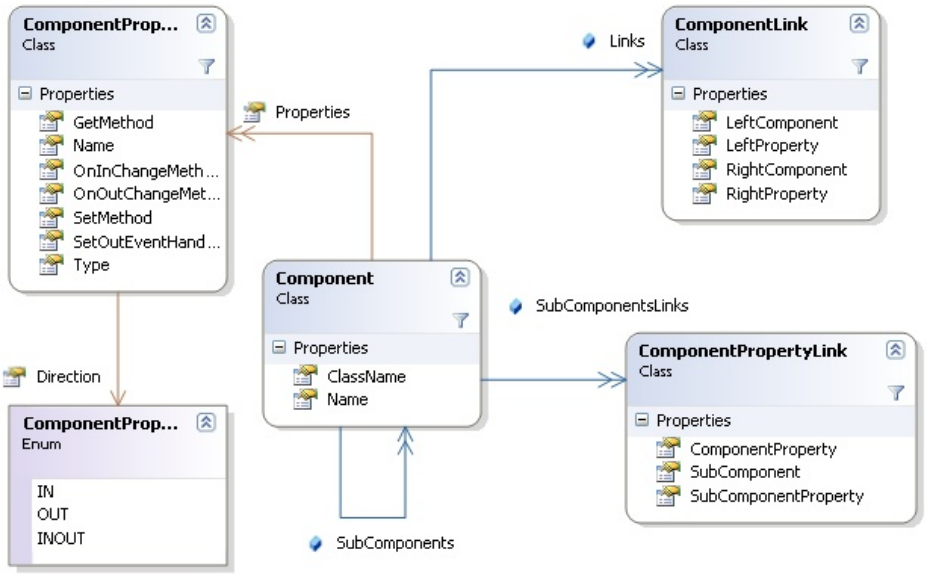

FIGURE 3. Independent Components Model

4. Research Results

4.1. **The developed framework for interoperability.** During this research a framework for interoperability between .NET and Java has been developed. Current version allows Java classes to be used in .NET and vice-versa. The general context is: there is a data type *RemoteType* on the developing platform *RemotePlatform* and it should be used in a class *C* on developing platform *ClientPlatform*. Based on the *RemoteType* an XML file is generated that specifies how the type can be used. Using this XML file the source code for the *ProxyType* is automatically generated. The *ProxyType* is written in the *ClientPlatform* programming language. This process is exposed in the figure 4.

4.2. **The Building Blocks Dev Studio.** Figure 5 presents a BBDevStudion print screen. The picture shows an application model with two subcomponents components: (1) UserInterface and (2) ALU. Both components have two properties: (1) Request and (2) Response. The data type of the Request property

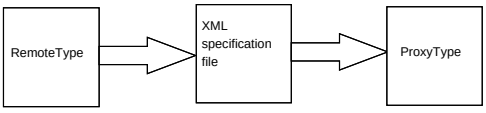

FIGURE 4. Automatically Generating the Proxy Type

is ALURequest and the data type of the Response property is ALUResponse. These two types are declared in a .NET dll file.

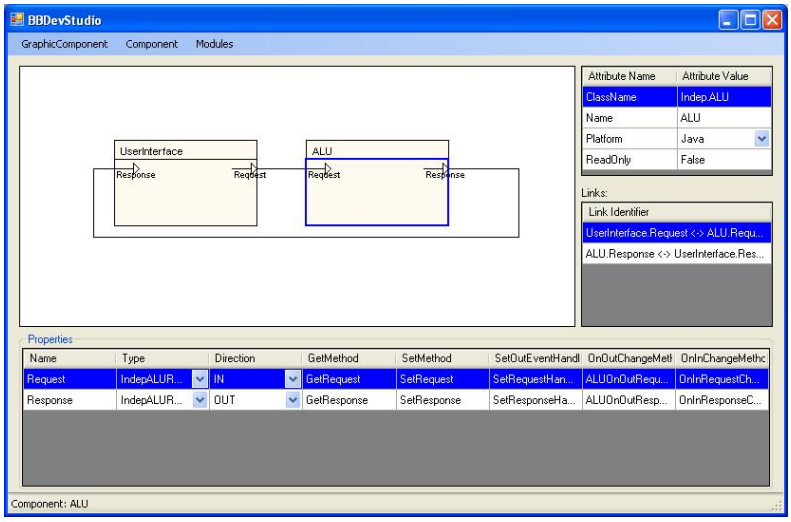

FIGURE 5. BBDevStudio Print Screen

The *UserInterface* component can use easily the *ALURequest* and *ALUResponse* data types because are written in .NET but the *ALU* component cannot use directly these data types, it can only use proxy types written in Java that can communicate with real data types written in .NET.

On the other hand, the parent component that contains the *UserInterface* and *ALU* components, can easily create the instance of *UserInterface* but should use a proxy to *ALU* for working with the real *ALU* component written in Java. The *ALU* component should use a proxy to the parent component in order to notify that a *Response* is ready to deliver to the *UserInterface* component.

ACKNOWLEDGMENT

The author wishes to thank for the financial support provided from programs co-financed by the Sectorial Operational Programme Human Resources

Development, Contract **POSDRU 6/1.5/S/3** "Doctoral studies: through science towards society".

The author would like to thank professor Bazil Pârv for his precious help.

References

[1] Paul Horatiu Stan, *A proposed technique for component based software development*, ZAC 2010, 52-56.
[2] Paul Horatiu Stan, Camelia Serban, *A proposed approach for platform interoperability*, Studia UBB 2010, 87-98.
[3] B Nolan, B Brown, L Balmelli, T Bohn, U Wahli *Model Driven Systems Development with Rational Products* IBM 2008.
[4] Edward L. Lamie *Real-Time Embedded Multithreading using ThreadX and MIPS*, Newnes Pap 2008
[5] David Chappell. *Introducing SCA*, DavidChappell and associates, July 2007, http://www.davidchappell.com/writing/Introducing_SCA.pdf last accessed on June 09, 2011.
[6] *SCA Service Componeent Architecture, Assembly Model Specification* 2007, http://www.osoa.org/display/Main/The+Assembly+Model, last accessed on June 09, 2011.
[7] Dominic Sweetman *See MIPS Run, Second Edition* Morgan Kaufmann; 2 edition 2006.
[8] Ingo Szpuszta, Mario Rammer, *Advanced .NET Remoting 2nd Edition*, APRESS February 2005.
[9] Bill Moore, David Dean, Anna Gerber, Gunnar Wagenknecht, Philippe Vanderheyden *Eclipse Development using the Graphical Editing Framework and the Eclipse Modeling Framework*, IBM 2004
[10] Stephen A. White. *Introduction to BPMN* 2004 http://www.bptrends.com/publicationfiles/07-04 WP Intro to BPMN - White.pdf, last accessed on June 09, 2011.
[11] Vincent Massol, *JUnit in Action*, Manning Publications November 2003.
[12] Clemens Szyperski, *Beyond Object-Oriented Programming*, second edition, ACM Press New York 2002
[13] Bruce Eckel, *Thinking in Java*, fourth edition, MindView, Inc, 2002.
[14] Ethan Cerami, *Web Services Essentials: Distributed Applications with XML-RPC, SOAP, UDDI and WSDL*, O'Reilly Media, Inc. February 2002.
[15] William Grosso, *Java RMI (Java Series)*, O'Reilly Media, Inc. Octomber 2001.
[16] Erich Gamma, Richard Helm, and Ralph Johnson, and John Vlissides, *Design Patterns: 50 specific ways to improve your use of the standard template library.* Pearson Education, Inc 1995.
[17] Model Driven Architecture, http://www.omg.org/mda, last accessed on June 09, 2011.
[18] http://www.w3.org/TR/soap, accessed on June 09, 2011
[19] http://en.wikipedia.org/wiki/Component-based_software_engineering *last accessed on June 09, 2011*

Babeş-Bolyai University, Department of Computer Science, M. Kogălniceanu St., 400084 Cluj-Napoca, Romania
E-mail address: horatiu@cs.ubbcluj.ro