AlterNative

July 20, 2012

0.1 Introduction

Nowadays all the service sectors like industrial, energy, healthcare, security within others are including new technological devices connected in the network.. The number of interconnected devices is increasing exponentially in the last years. Therefore there are appearing huge variety of devices, with different operating systems and different programming languages. In other words the future situation can be described as an heterogeneous distributed system. The future of the devices tends to converge in a multiplatform environment composed by different devices and elements but with similar capabilites and possibilities, not only smartphones, smart TVs or tablets but also all kind of elements.

In this work we present the AlterNative software as a cross-platform code translation tool focused on the performance of applications maintaining the maximum number of functionalities. AlterNative tries to solve the problem of multiplatform software avoiding the use of elements which can be perjudicial to the performance like virtual machines and aims to increase the efficiency of the code by using native languages.

0.2 Concept

The concept of AlterNative is to maximize the idea of Internet of Things by providing a tool for easy port applications from high-level languages (such as .NET) to native languages (such as C++). Most of the actual systems are C++ compatible, thus if the application is ported to this language, it can be executed in several platforms (i.e. smartphones, tablets, embedded systems, computers with different operating systems).

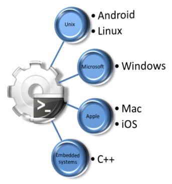

Figure 1: AlterNative concept

The philosophy of AlterNative is to provide the user with a system for taking the advantage of fast developing of high-level languages and also take the advantage of performance of low-level languages, and also exploit the possibility of run the native code in several systems, in other words, this philosophy is similar to the WORA (Write Once, Run Anywhere) slogan created by Sun Microsystem to illustrate the cross-platform benefits of the Java Virtual Machine. The difference is focused on the final performance since AlterNative outputs in native language and does not depend on any virtual machine.

0.3 Process

The process of the AlterNative software is divided in three parts: Decompilation, translation and compilation.

First of all the assembly received is passed through a decompiler in order to extract the source code. In this case the code extracted is C#. The code is not extracted in text format, but in an AST (Abstract Syntax Tree) that is an abstract representation with nodes and hierarchies of the code. This representation is organized in a tree from the top-level (Assembly) until de low-level (instructions, types and constants).

The step of translation consists on apply different conversions to the AST in order to obtain a second AST representing the source code but in other language (in this case C++).

The final step consists on compile the C++ code into a new assembly. This final assembly maintains the same funcionalities of the first assembly but takes benefit of the native code performance.

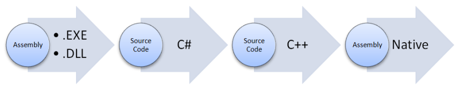

Figure 2: Process

The effort of this software is not focused on the code translation, but it is focused in maintain all the features and functionalities of the original code. The problem is that these features and functionalities often are directly related to the language, for instance a garbage collector, specific expressions or even language syntax. AlterNative is able to provide most of the features of the original code to the final code using external open-source libraries and propietary libraries in order to be fully compatible with high-level languages.

0.4 Scenario

The proposed scenario in this work is described in the Figure 3. The scenario is composed by an applications database, a Set-top box connected to a TV and the AlterNative service deployed in cloud computing. The applications database and the AlterNative service are separated into two different pieces because it is possible that the receiver device is compatible with the application distributed directly from the database.

The applications data-base contains executables compiled in high-level languages and depend on a virtual machine. In this case these languages are all the language based or compatible with .NET virtual machine (i.e. C#, VB.NET, Java). If the set-top box is not compatible with these languages, the executable is sent to AlterNative service in order to be converted. AlterNative converts the code into a native code (C++) and compiles it for the appropiate processor. The final application is deployed and executed in the Set-top box.

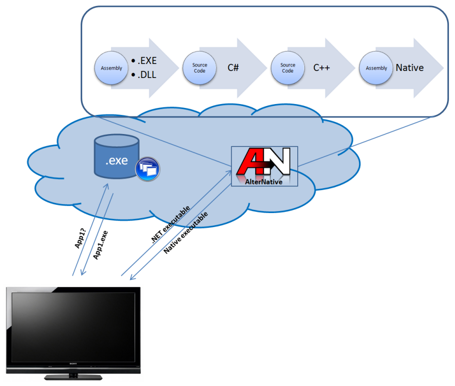

Figure 3: Scenario

This service can not only be applied to set-top boxes, but also for smartphones, tablets, computers or even embedded systems which are connected to the network. Furthermore, the most usual scenario will be a network with several devices with some computational power able to execute applications compiled for different operating systems, or processors. AlterNative improves this scenario compiling the applications for every device with a native performance and without the need of a Virtual Machine or the need of support the same operating system or the same processor.

0.5 Conclusions

Most of the cross-platform tools in the community are focused only on fast development for different devices (for instance games for smartphones). AlterNative a part of maintaining the objective of facilitate the easy and fast development, also is focused on the final performance of the application. Multiplatform systems based on virtual machines, or in HTML + CSS frameworks are useful for visual applications which do not require a high computational power, the main advantage of AlterNative software is that it is able to run code with native performance while the user is programming its code without taking into account the final application performance.