DOLPHIN-FEW – AN ARCHITECTURE FOR COMPILERS DEVELOPMENT, MONITORING AND USE ON THE WEB

Paulo Jorge Matos
ESTiG-IPB, 5300 Bragança, Portugal
pmatos@ipb.pt

Pedro Rangel Henriques
Universidade do Minho, 4700 Braga, Portugal
prh@di.uminho.pt

Abstract

DOLPHIN is a framework developed to help the construction of high performance, multi-language and retargetable compilers. It is constituted by a set of components, used to build and test new compilers or compiler routines; and by a set of tools, used to access, manage and develop the components. To improve and enlarge the functionalities of the DOLPHIN, several small projects were implemented around the framework, one of them is the DOLPHIN – Front-End for the Web, whose the goal is to build a set of applications to make available via web a set of services related with compilers, namely: an integrated development environment, a system for monitoring and analyze the behavior of the compilation tasks and an environment for the development of compiler components. This article presents the goals, the components and architecture of the DOLPHIN-FEW and the relation between this project and DOLPHIN framework.

Keys words: Compilers development, frameworks, web applications

1 Introduction

DOLPHIN is a framework [2] developed to help the construction of high performance multi-language retargetable compilers. It is constituted by a set of components used to build and test new compilers or compiler routines; and by a set of tools, used to access, manage and develop these components.

The development of the DOLPHIN framework is based on the idea that the compiler tasks can be classified as belong to the one of the three main compiler zones: the front-end, whose the most important goal is to interpret the source program; the back-end whose the main goal is to generate the output code; and the middle-level, responsible for all the other tasks, namely the ones that are independent of the characteristics of the language used to write the source program and independent of the characteristics of the computer architecture (processor + operational system) over which the output code should run.

The framework contains components that implement the several tasks of the compilation process, like: front and back-ends for the most well know programming languages and computer architectures; and a large set of code analysis and optimizations routines (that run over the middle-level). The compilers are built combining these components. The framework also includes components to measure the efficiency of the compilation tasks and tools to access, manage and develop new components (or to modify the ones that already belong to the framework).

Based on the framework, there were developed several small projects to accomplish specific goals. One of these projects is the DOLPHIN – Front-End for the Web (DOLPHIN-FEW), whose the main goal is to put available via web all the necessary services to work with the DOLPHIN framework,

specially to develop (implementation and test) full compilers, components for compilers and, work as a test-bed to teach topics about compilers construction.
DOLPHIN-FEW architecture is the main subject of this paper, which is divided in four sections: this one (the introduction); the second section that gives a brief introduction to the DOLPHIN project (framework and the main sub-projects); the third section that introduces the DOLPHIN-FEW and, the last one that presents the conclusion.

2 DOLPHIN project

Figure 1 shows the architecture of the DOLPHIN framework, including the sub-project DOLPHIN – Compiler Components Development System (DOLPHIN-CCDS) and the DOLPHIN-FEW. To guarantee the compatibility and to make easier the integration of the components, DOLPHIN defines a model of code representation, designated by DOLPHIN Internal code Representation (DIR), over which the several components work, including the front-ends, back-ends and the code analysis and optimizations routines.
For the subject of this paper it is important to explain that the framework contains a front-end that converts a XML description of the code representation into DIR (*DIRcnv*); and two back-ends: one to represent the data contained on the DIR into the HTML format (HG) and another, to generate the eXtended Markup Language (XML) representation of the DIR (XG).

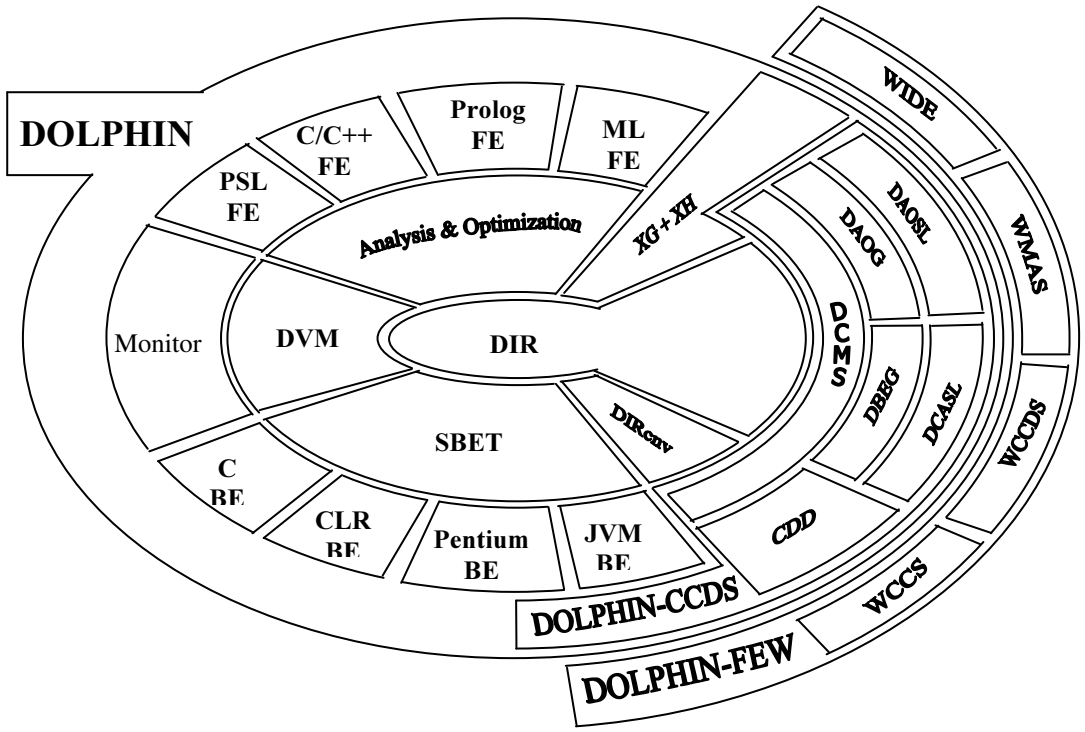

Figure 1 – DOLPHIN project architecture.

It is also important to give a brief explanation about the DOLPHIN-CCDS. This one supplies tools to manage and build new components to the framework, like the DOLPHIN Analysis and Optimizations Generator (DAOG) and the DOLPHIN Back-End Generator (DBEG) [1]. Both tools require a specification, which is done using, respectively, the DOLPHIN Analysis and Optimizations Specification Language (DAOSL) and the DOLPHIN Computer Architecture Specification Language (DCASL). It is also possible to build "by hand" new components, which is designated by Component Direct Development (CDD). Notice that most of the DOLPHIN-FEW services supply interfaces to the DOLPHIN-CCDS tools.

3 DOLPHIN-FEW architecture

DOLPHIN-FEW is project under development that intends to put available via web all the necessary services for the compilers development, which include: access to the DOLPHIN framework to build the compiler; tools to analyze and observe the compiler behavior; and an Integrated Development Environment (IDE) to make the final tests and to use the compiler. These services are implemented, respectively, by the Web Compiler Construction System (WCCS); by the Web Monitoring and Analyze System (WMAS); and by the Web IDE (WIDE). There is also a fourth service to manage the framework, which is designated by Web Compiler Components Development System (WCCDS) and a common graphical interface to integrate the several services. Each user has its own account where is saved the state of the environment, the specification of the compilers, the log files with the list of errors, the final product (the compiler), etc.

3.1 Web Compiler Construction System

The WCCS is the component used to create the projects. For each one, the user can consult and choose graphically the components of the framework that should be used on the compiler under construction and, if required, parameterize the components and specify the order by which they should be executed. In practice, this is done supplying a template that is fulfilled by the user. Some parts of the template are obligatory, for example it should contain a front-end and at least one back-end. The other parts are most of the times optional. Notice that the compiler can contain several back-ends and these ones have not to be specifically at the end of the compilation process (i.e. it is possible to generate the XML of the several stages of the compilation process).

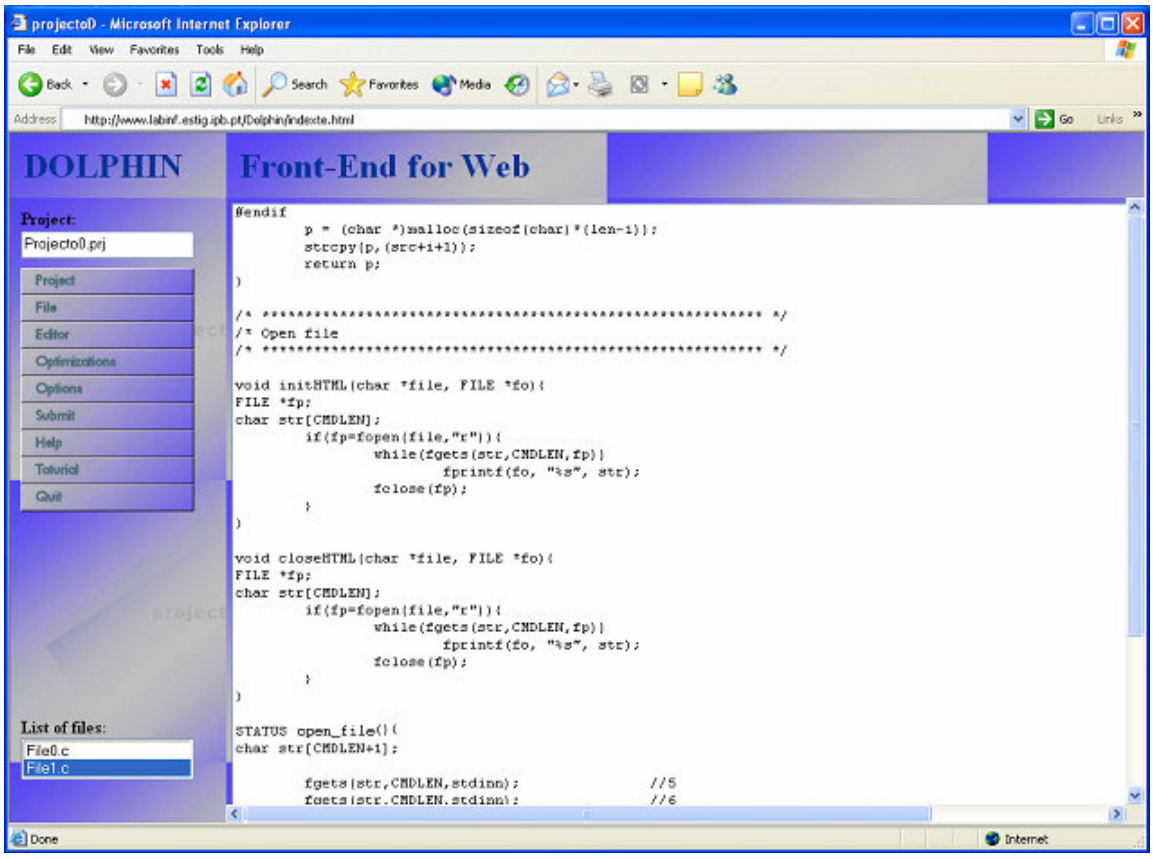

Figure 2 –DOLPHIN Web Integrated Development Environment.

There are also templates to implement specific components (that are not integrated on the framework), and to use them on the specification of the compiler structure. The WCCS, using the DCMS, guarantees the full construction of the compilers.

3.2 Web Integrated Development Environment

WIDE was initially built to put available via web a full environment for software development. The idea was to show the potentially of the compilers built with the DOLPHIN framework, allowing to edit and compile the source files and, to choose the type of output code or the optimizations that should be applied. But has also some particularities that make it unique, for example:

- Potentially, it can offer most of the technology available at DOLPHIN, namely: multi-language support, retargetable compilation and a large set of code optimization routines;
- It is an IDE available via Web, this means that can be used in any place that has a computer with access to the Internet;
- It works as an interface to set the DOLPHIN compilation options, namely: the code optimizations, the target computer architecture, the information that the compiler should generate, etc.

DOLPHIN-FEW integrates this IDE to supply an environment to test and use the compiler under construction, adapting, as long as possible, the environment to the characteristics of the compiler. For example, to select the code optimizations integrated on the compiler or to choose the stages of the compilation process that we want to inspect. Figure 2 shows the graphical aspect of the WIDE.

3.3 Web Monitoring and Analyze System

WMAS can be used, after the WCCS generates the compiler, to analyze and observe the execution of the compilation tasks. This is done integrating on the specification of the compiler structure, back-ends to generate HTML or XML [3]. Each stage that we want to monitor should have its own back-end. The use of these back-ends has consequences on the options available at the WIDE. If these options are activated, the WIDE automatically calls the WMAS, when a source program is submitted to the compilation, to visualize the information.

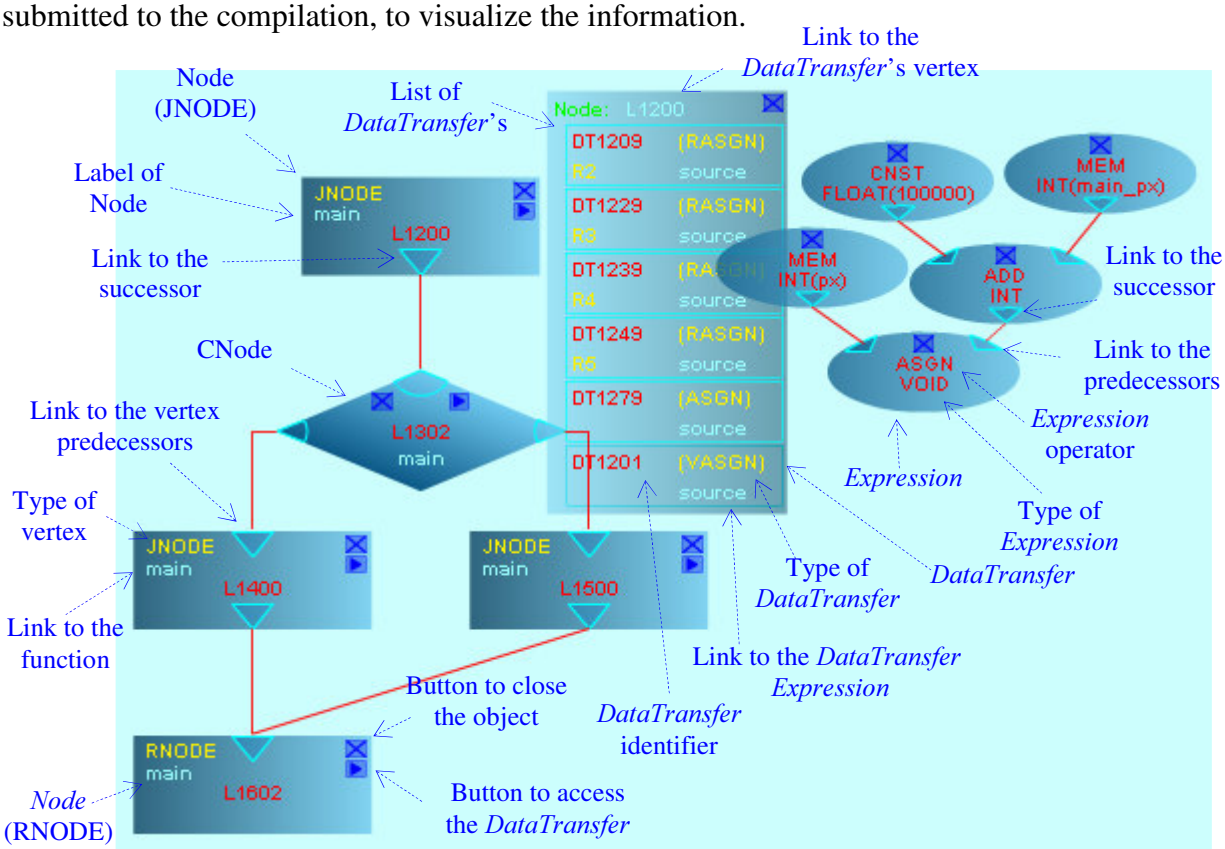

Figure 3 –An example of the visual representation built by the WMAS based on the XML.

If the chosen back-end produces HTML, the DIR is visualized with a simple web browser, but if the back-end produces XML, then the information is visualized with a tool built with Macromedia Flash[1], that uses the XML to dynamically construct a graphical interactive representation of the DIR. Figure 3 shows the graphical aspect of the representation produced by the Flash application.

3.4 Web Compiler Components Development System

The WCCDS is the service used to access the tools of the DOLPHIN-CCDS, like the DBEG and the DAOG. In practice, WCCDS should supply context sensitive editors to write the specification, of the computer architecture and the code analysis and optimizations routines. But the WCCDS will also be the service used to manage the framework, allowing to: insert (components registration), actualize or remove the framework components. The WCCDS should also be used to make the control of the user's accesses, each one should have its own instance of the framework to be free to change, remove or insert new components (or at least the state of the framework).
Unfortunately the implementation of the WCCDS is quite dependent of the results of the project DOLPHIN-Compiler Components Development System, which still under construction and, as consequence, had delayed the implementation of the WCCDS. But the plan is to build a graphical interface with context sensitive editors to:

- Help the edition of the several specifications, eventually, combining them into a single specification;
- Supply templates of some of these specifications (i.e. the computer architecture specification of known processors);
- Represent and work with the framework components like visual objects;
- Write the DCMS specification using graphical objects;
- Develop new extensions to the DAOG, such as parametric routines (represented graphically);
- Automatically integrate the components into DOLPHIN framework;
- Put the components immediately available to be used.

4 Conclusion

The goal of this paper was to introduce the architecture of DOLPHIN-FEW. The idea around this architecture is quite novel (we do not know nothing similar). We dare to say that is not one more solution. DOLPHIN-FEW takes advantage of some web technologies to give a new perspective and possibilities for the compilers development, proposing a new set of web services.
All the components of DOLPHIN-FEW have space to grow, particularly the WMAS and the WCCDS that should produce excellent results and, eventually, some new ideas. Probably, the principal obstacle of DOLPHIN-FEW is the DOLPHIN framework itself that still have many things under implementation and some that are only on the paper. But we believe that now, that the results started to appear, the development time will decrease.
Along the paper many of the future goals of this project were introduced. We still work on the DOLPHIN project and especially on the DOLPHIN-FEW to achieve these goals, hopefully to present soon more results.

References

[1] Matos, P. (1999) Estudo e desenvolvimento de sistemas de geração de back-ends para compiladores. Master Thesis, Universidade do Minho.
[2] Matos, P. (2002) *The DOLPHIN framework*. Technical Report, Universidade do Minho.
[3] Matos, P. and Henriques, P. (2003) DOLPHIN-FEW – An example of a web system to analyze and study compiler behavior. Proceedings of the E-Society Conference.

[1] Flash is a registered trademark of the Macromedia, Inc.