DOLPHIN-COMPLAB: A VIRTUAL COMPILERS LABORATORY

PAULO J. T. MATOS

Departmento de Informatica e Comunicações, Politécnico de Bragança,
Campus de Santa Apolónia, 5300 Bragança, Portugal.
E-mail: pmatos@ipb.pt

PEDRO R. HENRIQUES

Departamento de Informática, Universidade do Minho, Gualtar 4700 Braga, Portugal.
E-mail: prh@di.uminho.pt

The goal of this paper is to present the Dolphin-COMPilers LABoratory (COMPLAB), an integrated environment conceived for the investigation, teaching, development and use of compilers. It employs Web technologies to make available a virtual laboratory containing a set of functionalities to: implement, analyze, test, evaluate and simulate compiler routines. This environment aims to support the classes of Language Processors of the graduate and post-graduate levels of the Computer Science courses, inside and outside of the classroom. The environment, reachable using a simple Web browser (connected to the Internet), should supply three distinct levels of access that correspond to three distinct types of users: conventional users, that can access to the laboratory to work on the projects where they are registered using the tools supplied by the DOLPHIN-COMPLAB; the project coordinators responsible for the management of the project (define the goals, choose tools and components, manage the users of the project, …); and the main coordinators that are responsible for the laboratory (registration of projects, integration of components, laboratory cleanness, etc). A conventional user has access to a several tools that support the two main stages of the compilers development process: the implementation and test. The functionalities supplied for the implementation allow to develop, modify and change the compiler components and the compiler structure. To support the test and evaluation of the compilers, it is supplied an integrated development environment where the users can analyze the behavior of the several compilation tasks and measure the performance of the compiler.

1 Introduction

Dolphin-COMPilers LABoratory (COMPLAB) is a project to build an integrated environment for investigation, teaching and development of language processors, namely for compilers. The main idea is to use some Web technologies to make available a virtual laboratory to supply a set of features to support the development process of compilers and compiler components. DOLPHIN-COMPLAB was conceived to satisfy the necessities of the compiler developers, investigators, teachers and students, at the different stages of compilers development process, namely at: implementation, test, analysis behavior, performance measure and simulation.

DOLPHIN-COMPLAB is part of a bigger project, conceived to promote the development of compilers and compiler components, which is designated by DOLPHIN System [1]. DOLPHIN-COMPLAB is essentially an integrated environment accessible via web that joins many of the features available on the DOLPHIN System, namely several tools and a framework that contains the components used to build and test the compilers.

This paper is organized in four sections: this one where is introduced the subject of the paper; the next section where is done a brief introduction to the DOLPHIN System, explaining the type of tools supplied by this system and the relation among them; at the third section, it is presented the architecture conceived for the DOLPHIN-COMPLAB; and last section where is done the conclusion.

2 DOLPHIN System

DOLPHIN System, represented at the Figure 1, was built based on the DOLPHIN framework, which is essentially a set of component (compiler routines), that work over a specific form of code representation - the DOLPHIN Internal code Representation (DIR). These components can be combined to construct new compilers. Over the framework works the DOLPHIN-Framework Management System that allows to manage and control the access to the framework. Around this central core, work several other sub-systems,

each one dedicated to a specific part of the process that goes from the development of new framework components to the use of the compilers built with the DOLPHIN System. Some of these sub-systems are used to support the DOLPHIN-COMPLAB, it is the case of: DOLPHIN-Web Integrated Development Environment (DOLPHIN-WIDE); DOLPHIN-Compilers Development System (DOLPHIN-CDS); and the DOLPHIN-Framework Components Development System (DOLPHIN-FCDS).

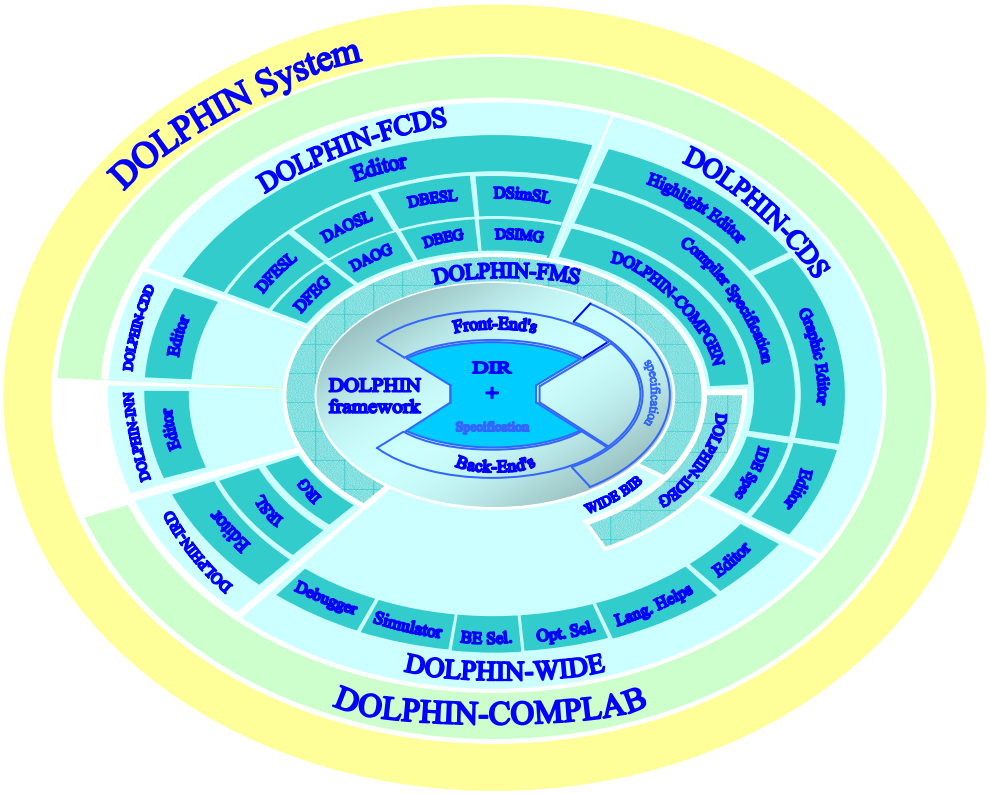

Figure 1. - Architecture of the DOLPHIN System.

DOLPHIN-WIDE was built as a show-case of the technology implemented on the DOLPHIN framework [2,3]. It provides all the essential things for the software development, such as: projects management, highlight editors, debugger and simulators. But the main part is a multi-language retargetable compiler, constructed with the solutions implemented on the DOLPHIN framework that allows to compile code from different languages and to generate code for several computer architectures. But the construction of the DOLPHIN-WIDE gave the necessary knowledge to build a library – WIDE LIB- that is used by the DOLPHIN- IDE Generator (DOLPHIN-IDEG), to automatically generated IDE's for the compilers built with DOLPHIN System (more precisely with the DOLPHIN-CDS). Notice that this is a very important feature since allows to construct simultaneously the compiler and environment to use the compiler.

DOLPHIN-CDS is the IDE that allows to build compilers using the components of the framework. The user only has to specify the structure of the compiler, which means choose and parameterize the components and determine the execution order. Based on the specification, that could be done textually or graphically, the DOLPHIN-CDS automatically builds the compiler. It is also with this system that the user can specify the characteristics of the IDE for the constructed compiler and invoke the DOLPHIN-IDEG.

Since it is not feasible to think that DOLPHIN framework can supply all the possible components, it is important that DOLPHIN System supplies the necessary features to let and promote the development of new components by the users. As a matter of fact, this is one of the main principles behind the conception of the DOLPHIN System, the user is incited to implement new components and submit them to be included on the framework. To accomplish these necessities, it was conceived the DOLPHIN-FCDS that supplies several tools to build components, namely: front-ends to make the interpretation of the source language to the DIR; back-ends to generate the output code; and code analysis and optimizations routines. But also tools to build

the highlight editors and the simulation environments used on the IDE produced by the DOLPHIN-IDEG. It is always possible to the user, develop new components without use these tools, which is designated by DOLPHIN-Components Direct Development (DOLPHIN-CDD).

DOLPHIN-COMPLAB was conceived to integrate in a single environment all presented tools and some extra features that allow to supply the necessary services and organization to attract the compiler developers and users community (compiler investigators, developers, teachers and students and, also software programmer), and simultaneously promote the development of the DOLPHIN System.

3 Architecture of DOLPHIN-COMPLAB

Now that the main goals and the type of solutions that are implemented at DOLPHIN System were introduced, it is time to explain how the services of the DOLPHIN-COMPLAB are organized.

To sustain the conception and guide the development of the DOLPHIN-COMPLAB, it was established several rules. First rule was: the users should work over an instance of the main framework, designate simply by "framework". Notice that if the users work directly over the main framework, they can cause unrecoverable damages to components and even to the framework. Second rule: for each new project it should be created a new instance of the framework and should be settled a responsible for the project. Third rule: it should exist the figure of "project coordinator" (the responsible for the project), that could be, for example, a teacher or a supervisor of an investigation team. The project coordinator should manage the users that have access to project (insert, remove and change the user's settings); define the characteristics of the project, like: goals, project scheduling, utilization politics of the components, etc; negotiate with the central system to request new components or to submit components to be inserted on the main framework; and, of course, maintain operational the framework and all the other functionalities associated to the project. Fourth rule: it should be supplied an interface to the conventional users, over which they can access to the shared elements of the project, like the framework, tools, libraries, shared specifications, and also to some applications (white-board, log-book, instantaneous messages, mail list, etc). The user should also be able to personalize the interface and maintain personal documentation. Fifth rule: DOLPHIN-COMPLAB should be support by a main system, managed by the main coordinators that are responsible for: the creation and elimination of the projects; maintenance of the main framework (insert, remove and actualize the components, manage the components versions, etc); deal with the several projects coordinator (negotiate components, deliver information, remove doubts, etc), etc. Considering all these goals we defined three profiles: main coordinator, project coordinator and conventional user, each one with its own interface.

The information about the users (login, password, email and other information) is maintained on a database that contains also the identification of the users that have privileges of main coordinators and the information about the projects (identification of the project coordinator, the components and tools requested for the project, the users of the project with the correspondent privileges and environment settings). The information about the components and tools supplied by the DOLPHIN-COMPLAB (description, helps, versions, source and binary code, list of problems, specification, etc) is also maintained on the database.

Files, like the binary and source files of instantiated framework, the specifications described by the users, the components implemented without the tools supplied by the DOLPHIN-COMPLAB, and other personal user files, are maintained on the accounts of the projects or on the accounts of the users. Notice that the users can create, read and write files to the accounts but for security reasons they can not execute any other file that is not supplied by the DOLPHIN-COMPLAB. Even the compilers built by the users, can be executed only inside the IDE generated by the DOLPHIN-IDEG. For the same reasons, each project can have only a limited number of compiled files (produced by the compilers built with the DOLPHIN system). When the limit is exceeded, the oldest files are automatically deleted. These files can not be executed directly on the user's accounts, but can be downloaded. The same politician is applied to the IDE's built with the DOLPHIN-IDEG, each project can have only a limited number of available IDE's. The tools and other applications supplied by the DOLPHIN-COMPLAB are shared by the several projects, but the users can personalize the interfaces.

The users can access to the DOLPHIN-COMPLAB services via the web page supplying the login and password. After the validation it is showed a list with the projects where the user participates and, if it is the

case, a link to the Web page of the main coordinators. The links can be to project where the user is the coordinator or project where the user acts as a conventional user. Notice that a project coordinator is automatically joined to the list of the conventional users of the correspondent project. Next three sections describe the interfaces conceived for each of the three types of users.

3.1 Main coordinator interface

The main coordinators are responsible, by one side, for the management of the framework, which includes the components, tools and applications (generically designated by "element"); and, by the other side, for the management of the projects. The interface supplied for the main coordinators allows to execute all necessary tasks without have to deal directly with the operator system or with the database server.

Insert a new element deploys a message to all project coordinators with the information about the element. Actualize a element forces a message to the users of this element, with the information about the changes and, if the case, with warnings about the oldest versions. Remove an element is a little more complex, if it is a framework component, the users are advised of the discontinuation of the component, eventually with the explanation of why the component was removed and with information about alternative solutions. Notice that the users still access to the component since they work with an instance that is installed on the project account and not with the original component. If the element is an application or a tool, the users are first warned about the intention of remove the tool, then after a certain period of time and if no project coordinator claims for the use of the tool, this one is removed.

To insert a new project it is necessary to: assign a project coordinator, that is immediately add to the users of the project; and choose the components, tools and applications that should be available for project. Automatically, it is created a new account for the project (and if necessary for the project coordinator), where is deposited an instance of the framework with the chosen components. All the other users of the project should be inserted by the project coordinator. When a project is created, it is automatically defined a mail list for the project users. Remove a project consists on remove the information about the project from the database and them remove the project account. Notice that the general information about the users is not removed from the database (the main coordinators check periodically for users that do not belong to any project to remove them).

3.1.1 Project coordinator interface

The project coordinator is the responsible for the management of the project. He can insert and remove the users or change the profile of each one to control the access to the components and tools. There are four attributes for the components: "read", "write", "use" and "hide". The tools only have the "use" and the "hide". Controlling these attributes, the project coordinator can give or remove the access to the components and tools.

When is inserted a new user, it is created a register into the database and an account for the user. The information about the development environment of the user and about the privileges of the user is saved together with the information of the project. Remove a user from a project, consists on remove the information about the user from the information about the project and the correspondent account.

The project coordinator is also responsible for the maintenance of the project account and of some services, like the white-board, the log-book, the scheduler and other services that still under study. Eventually, he can control the access to the mail list and to the instantaneous messages. The log-book maintains the information about the operations that are executed on the project, which is done automatically and separately for each user. But the project coordinator can remove or add new entries to the log-book or generate reports based on the log-book. It is also of the responsibility of the project coordinator to define the scheduling of the project. Messages are sent automatically to the users requesting information about the execution of the scheduling. The project coordinator can even define the tests that should be executed for each task or the information that should be supply by each user.

3.2 Conventional user interface

It is the conventional user (or simply "user) that labours with the framework and tools to implement new components or compilers. He has access to several applications that help to increase the productivity, like: the white-board to pass messages to the other users or write notes; the log-book where are registered all the operations executed over the framework; the scheduler to manage the development of the project; the instantaneous messages to communicate on real time or the mail list for off-line communication.

There are five main tasks that the user can execute: develop or change components; build compilers; generate IDE, test and use compilers. Each of these main tasks is supported by a specific set of tools. The development of new components is preferentially done with the DOLPHIN-FCDS. The user has to describe the component behavior/characteristics as a specification that is later submitted to the DOLPHIN-FCDS, producing a set of C++ files with all necessary routines and definitions to build the component. It is also possible to implement components directly without use the DOLPHIN-FCDS, which is designated by DOLPHIN-CDD. To test the component it is necessary to build a compiler that uses this component, but first the component must be registered on the project (this is one of the operations that is saved on the log-book). The compilers can be built submitting the specification of the compiler structure to the DOLPHIN-COMPGEN, a tool of DOLPHIN-CDS. Then the user has two options: download the compiler and test it personally or use the DOLPHIN-IDEG to generate an IDE to use or test the compiler. This can be done using the compiler specification and a second specification, where the user defines the features of the IDE, which can be specially prepared to be used for tests (with measure components and benchmark test); or to be used for development of software. The specifications can be shared registering them on the project. The project coordinator can request a report with evaluation parameters of the component/specification to accept the registration (defined on the project profile).

4 Conclusion

It is not yet possible to measure the utility, efficiency and productivity of DOLPHIN-COMPLAB because it is not finished (many things still under construction). Even other tools of DOLPHIN System, that support the DOLPHIN-COMPLAB, have many things that are not yet implemented. But the way how DOLPHIN-COMPLAB was planned, the principles that are behind the conception of the DOLPHIN-COMPLAB and more generically behind the DOLPHIN System, and the diversity of services that will be supplied, give us good expectation about this project, which we believe will be specially useful and auto sustainable for compiler investigators, developers, teachers and students.

SUIF and Zephyr [4,5] are conceived to support the development of compilers. But as long as we know, none of these solutions supply via Web an integrated environment that fully supports the development of compilers, like the DOLPHIN-COMPLAB. This makes, by one side, impracticable any type of comparison and, by the other side, proves the novelty of the ideas behind DOLPHIN-COMPLAB. As soon as we have a stable version of the DOLPHIN-COMPLAB, it is our idea to test it on the Language Processors classes of the graduate and post-graduate levels of the Computer Science courses.

References

1. Matos, P. and Henriques, P. *DOLPHIN framework*. Technical report (Universidade do Minho, Portugal, 2002).
2. Matos, P. and Henriques, P. *DOLPHIN-FEW - An example of a Web system to analyze and study compilers behavior*. On the International Conference e-Society (Lisbon, Portugal, 2003) pp. 966-970.
3. Matos, P. and Henriques, P. *A solution to dynamically build an interactive visualization system to the DOLPHIN-FEW*. Proceedings of the International Conference on Visualization, Imaging, and Image Processing (Spain, September, 2003) pp. 868-873.
4. Aigner, G. et al. An overview of the SUIF2 Compiler Infrastructure. Technical report (Computer System Laboratory, Stanford University).
5. Appel, A., Davidson, J. and Ramsey, N. *The Zephyr Compiler Infrastructure*. Technical report, 1998.