USING WWW TO IMPROVE SOFTWARE DEVELOPMENT AND MAINTENANCE: APPLICATION OF THE LIGHT SYSTEM TO ALEPH PROGRAMS

A. AIMAR, M. AIMAR, A. KHODABANDEH,
P. PALAZZI, B. ROUSSEAU, M. RUGGIER

docsys@ptsun00.cern.ch
Programming Techniques Group, ECP Division

M. CATTANEO, P. COMAS ILLAS

ALEPH Experiment

CERN, 1211 Geneva 23, Switzerland

Programmers who develop, use, maintain, modify software are faced with the problem of scanning and understanding large amounts of documents, ranging from source code to requirements, analysis and design diagrams, user and reference manuals, etc. This task is non trivial and time consuming, because of the number and size of documents, and the many implicit cross-references that they contain. In large distributed development teams, where software and related documents are produced at various sites, the problem can be even more severe. LIGHT, LIfe cycle Global HyperText, is an attempt to solve the problem using WWW technology. The basic idea is to make all the software documents, including code, available and cross-connected on the WWW. The first application of this concept to go in production is JULIA/LIGHT, a system to convert and publish on WWW the software documentation of the JULIA reconstruction program of the ALEPH experiment at CERN, European Organisation for Particle Physics, Geneva.

1 Introduction

Mastering the life cycle of large software projects implies the generation of many documents by different people working in several sites. These documents are produced at each stage of the life cycle process: requirement specifications, analysis documents, modelling diagrams, data dictionaries, source code, implementation notes, end user documentation, etc. For a programmer, the effective use of such documents is not easy because of their:

- **number and size**
  A software project typically generates tens of documents. In order to apply a change in the software, a programmer needs to read and understand many of them.
- **complexity**
  Software documentation and source code are heavily structured and cross-referenced. They are also interdependent since each of them refines statements decided in previous stages of the life cycle. This connectivity is only implicit: in order to understand a software, the programmer has to navigate among several manuals and

files, and to locate relevant information in each one of them. Typically, the programmer must navigate:

- from a function call to a library manual,
- from a variable to its declaration,
- from a language keyword to the language reference manual,
- from a data structure to its design diagram,
- from a software requirement to one or more user requirements,
- etc.

- **distributed production**
  Large software projects gather tens of developers, often spread across several sites in different countries. This complicates the management of source code and documentation; access to documents that are up to date and in step becomes more difficult.

These three properties of the software documents: high volume, strong connectivity and distribution are good justifications for using a publishing solution based on the World-Wide Web. The basic idea of LIGHT (LIfe cycle Global HyperText) [3] is to make all the software documents[1], including code, available on the WWW, with all cross-references automatically established. For instance a click on a subroutine call in Fortran code takes you to its documentation, a click on a data element leads to the corresponding data definition, etc. We have applied this concept to the JULIA reconstruction program of the ALEPH experiment at CERN.

2 The JULIA Software Documentation

The JULIA reconstruction program itself consists of a large Fortran 77 program (110K lines) and a User Manual. But, like all other offline software of the ALEPH experiment [4], the JULIA program relies upon general ALEPH and CERN models, packages and services, listed below.

- The *ALEPH data model* is a formal description of the physics data manipulated by the ALEPH programs. This description is based on the entity-relationship model and is written in *ADAMO Data Definition Language* (DDL). The ALEPH Data model consists of 160 DDL *subschemas*.
- The *BOS memory management system* is used to implement and manage in Fortran the data structures corresponding to the ALEPH data model. Elementary data structures are called *banks*. ALEPH uses more than 1,000 such banks.
- The *ALEPH database* contains detector and physics parameters accessed by all ALEPH programs.
- JULIA exploits two large libraries of routines: the *ALEPH library* (ALEPHLIB) and the *CERN library* (CERNLIB).

1. In the framework of LIGHT, a document is a file or a set of files that can be considered logically as a whole. This definition covers not only files produced with word processors, but also source code of programs, listings and data files.

- JULIA, like all other ALEPH programs, uses Fortran programming conventions that allow dedicated scripts to expand the code, extract documentation and variables lists from the source code.

This means that, in order to be used and maintained, the JULIA software requires a large and complex documentation in the form of paper (~800 pages of manuals), files (~4000 pages of listing) and databases. This documentation covers not only the JULIA program itself, but also the general ALEPH and CERN software and models used by it. As a consequence the JULIA/LIGHT system has to convert and/or link in the same web not only the JULIA documentation but also the general ALEPH and CERN documentation.

3 JULIA/LIGHT at Work

With JULIA/LIGHT, ALEPH programmers, documentation writers, maintainers of the data model and end user physicists are able to view through the web the entire Fortran source code, data definition, data design diagrams, as well as the JULIA, ALEPHLIB, CERNLIB, ADAMO and BOS manuals. All these documents are connected with hypertext links in such a way that, simply using the mouse, it is possible to perform helpful inspection. Figure 1 is a “navigation map” that shows the documents available on the web and their hypertext connectivity.

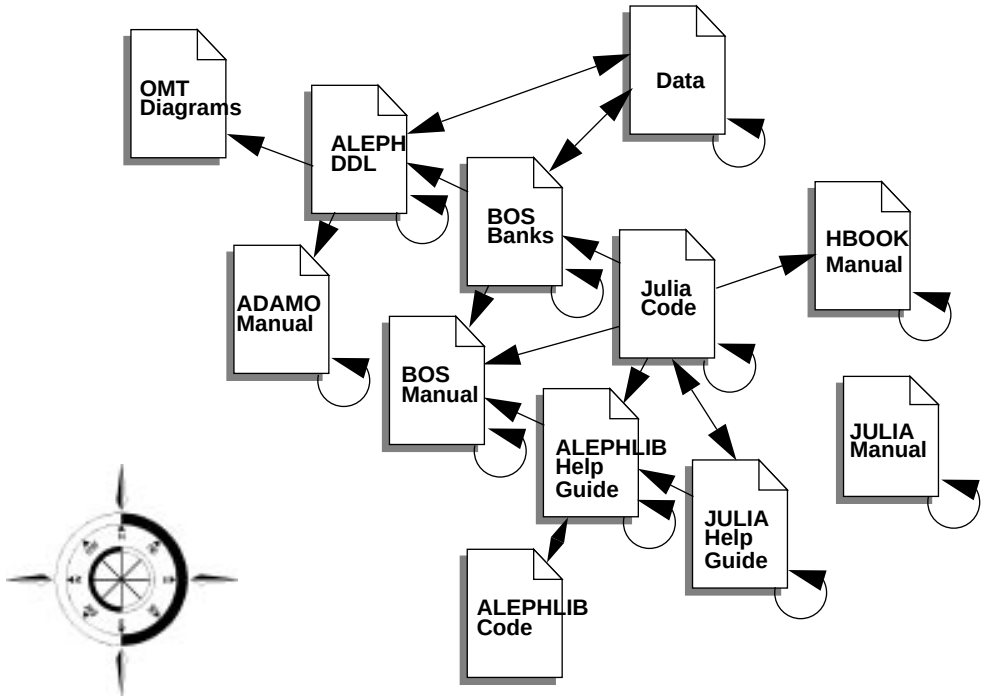

Figure 1: The JULIA/LIGHT navigation map shows the various JULIA software documents available on the web, and their connectivity. For instance, it is possible to navigate from the Julia Source Code to the BOS Banks descriptions.

For instance, from the code of a JULIA subroutine it is possible to navigate to:

- the source code of calling and called subroutines;
- the ALEPH data definition files and design diagrams;
- ALEPH Library manual and source code, and more.

Hyperised indices and dependency trees help the navigation through the various documents. Figure 2 gives an example of how a Fortran subroutine appears with the NCSA mosaic WWW browser.

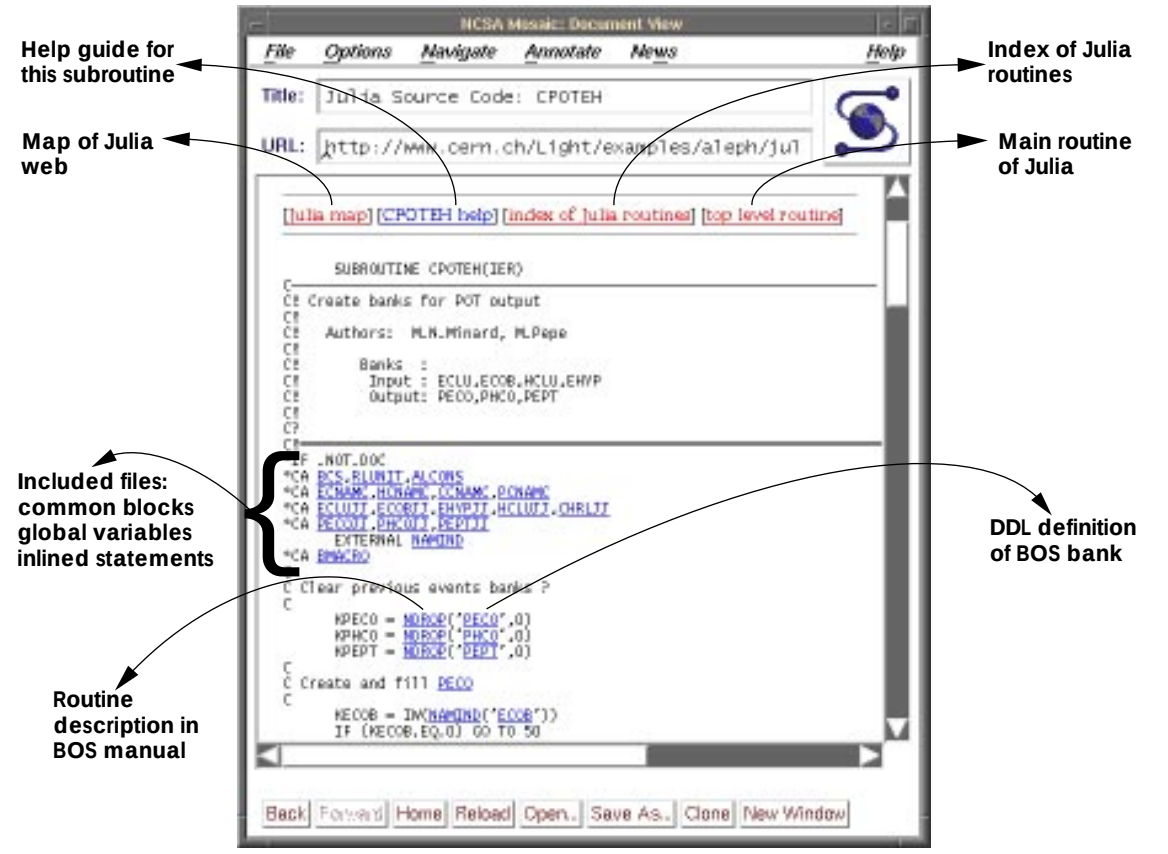

Figure 2: The source for a Fortran subroutine of JULIA, as seen with NCSA Mosaic. Underlined words correspond to sources of hypertext links. Depending on the kind of include file, clicking on a file name can take you to common blocks or to a bank description. Clicking on a routine name leads you to the corresponding documentation. At the top of the page, a navigation panel to related documents is provided.

The JULIA/LIGHT web is publicly accessible from the URL:

```
http://www.cern.ch/Light/
```

The final production version of the web will be installed end of September 1995. It will consist of approximately 6,000 HTML pages with 150,000 hypertext links. Until this date, a demonstration version is available.

4 Design and Implementation Issues

It is relatively easy to generate an HTML version of a document in a given format; plenty of such filters and utilities are available on the Web, for formats that are well-known as well as for others that are more exotic. In the framework of the LIGHT project, we have developed such converters for FrameMaker [7], Fortran 77, ADAMO DDL and OMTool Diagrams, as well as using the already existent converter for LaTeX [5]. It is nevertheless much more difficult to design a system that integrates these converters and provides the flexibility and power needed in large software projects such as JULIA. Four essential requirements have dictated the design of the LIGHT system:

- **Genericity**: the ability to easily add new converters.
- **Automatic cross-connection**: the ability to automatically connect several documents with hypertext links.
- **Configurability**: the ability to configure the connectivity, depending on the kind of application.
- **Incremental update**: the ability to reconvert only documents that have changed, without regenerating the web of all other documents.

These requirements have resulted in the architecture presented on Figure 3.

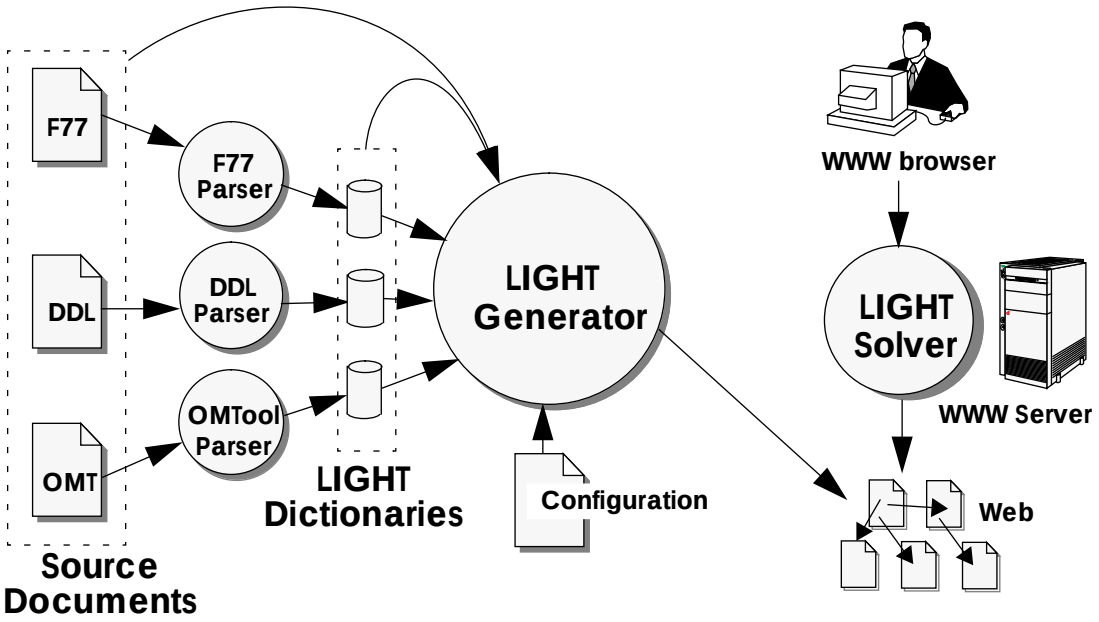

Figure 3: The Architecture of the LIGHT System. The conversion of source documents is performed in 2 steps. During the first step, source documents are analysed by specialised parsers that produce LIGHT dictionaries. LIGHT dictionaries contain the list of all possible hypertext sources and targets. The HTML generation is performed when all LIGHT dictionaries are available. During this step, the LIGHT generator reads source documents and generates the corresponding HTML. It creates hypertext links by navigating LIGHT dictionaries and associating targets with destinations, according to connectivity rules described in the configuration file. The resulting web does not contain a single URL; links are expressed through requests to the LIGHT Solver, which is a CGI program that is able to dynamically resolve the logical information contained in the request into a physical URL. This architecture enforces the 4 main requirements of the LIGHT system: genericity, automatic cross-connection, configurability and incremental update.

4.1 Genericity

In order to be reusable for software projects other than JULIA, the LIGHT system allows easy addition or interchange of converters. It is therefore necessary that converters are independent, e.g. a Fortran converter must not generate hypertext links to the Reference Manual of a library because this means that it will not be reusable in the context of other projects using different libraries. Under this constraint, how does one generate hypertext links across HTML documents produced by different converters?

4.2 Automatic Cross-Connection

We solve this problem through a 2-step conversion scheme.

- In the first step, the source documents are analysed by specialised parsers, one per document format. Each parser leaves its source document unchanged and produces a database, called a LIGHT dictionary, containing a description of all potential sources and targets for hypertext links (we call them tokens) in the source document. For example, the Fortran parser creates tokens for variable declarations and usage, for subroutine calls, for include files, etc.
- The second step starts when all source documents are processed and is performed by the LIGHT Generator. This generic program reads all LIGHT dictionaries resulting from the first step. It then converts the source documents one by one to HTML and establishes hypertext links by matching hypertext sources with potential hypertext destinations, as specified in the dictionaries.

4.3 Configurability

There are several ways to interconnect documents in a LIGHT web. For instance a routine call in a Fortran program may be linked to the corresponding description in the reference manual, or to its source code, or to a call tree, depending on the kind of application. Configuring the LIGHT generator to produce the required connectivity becomes thus an important issue. The LIGHT generator is driven by a configuration program containing rules such as:

```
In MODULE “JULIA Source Code”, link
TOKENS of NAME Routine and of TYPE “Routine Call” to
TOKENS of NAME Routine and of TYPE “Routine Description”
in MODULE “HBOOK Reference Manual”
```

The above rule specifies that routine calls in JULIA source code must be linked to the corresponding routine descriptions in the HBOOK manual.

4.4 Incremental Update

In the WWW, hypertext links are expressed with Uniform Resource Locators (URLs). These are, however, not appropriate to express the connectivity amongst several documents, as even small changes in the structure or location of one of the documents

may result in broken links in other documents [1]. For instance, if a reference manual is changed, links from other documents may become invalid, thus requiring that all other documents are reconverted to adapt to the new version of the reference manual. This is not acceptable for large software projects, where the reconversion time can be very long and new versions of any one document may be frequent.

Our solution relies upon the fact that, in a LIGHT web, each possible hypertext target has a type and a name (e.g. subroutine RECONS). The combination of a document, a target type and a target name identifies the target in unique way, and is less subject to changes than a URL. Thus, in LIGHT webs, hypertext links are expressed as requests of the form (document, name, type), instead of as URLs. A CGI program, called the LIGHT Solver, resolves this request to the corresponding URL and returns the relevant HTML file.

5 Status and Further Developments

The JULIA/LIGHT system is currently under development. The first production version is planned for the end of September 1995. Other longer term collaborations with ALEPH concern interactive access to physics data and analysis tools through the WWW. The application of LIGHT to other physics experiments is also foreseen, implying the development of new LIGHT parsers (e.g. C/C++).

More and more software projects in HEP are using software development standards, (e.g. ESA PSS-05 [6], object-oriented methodologies (e.g. Booch [2] and OMT [8]). These standards and methodologies encourage the production of requirements, analysis and design documents that increase the size and complexity of the software documentation. Another aspect of LIGHT is concerned with the development of templates and tools to allow the conversion and interconnection of such documents on the WWW.

Finally, we are considering ways to couple the LIGHT system with software development tools such as source code managers, e.g. CVS, Programming Environments, e.g. SNiFF, and CASE Tools, e.g. Rational Rose. The LIGHT system would then extract information directly from these tools and format it in HTML “on the fly”.

6 Conclusion

The LIGHT concept is an effective solution to deliver software documentation locally or world-wide, with all the added benefits of hypertext links. The JULIA/LIGHT system automatically converts and cross-connects all the JULIA software documents, on WWW. The feedback to early versions from ALEPH users shows that it significantly improves the readability and navigation of the JULIA documentation. The LIGHT software provides a generic scheme to build and plug in new converters and is thus extensible to other formats and programming languages. Maintenance of generated webs is facilitated with the possibility to update incrementally.

7 References

[1] A. Aimar, I. Hannell, A. Khodabandeh, P. Palazzi, B. Rousseau, M. Ruggier, J. Casey, N. Drakos, WebLinker - A tool for managing WWW cross-references. Proceedings of *Second International WWW Conference*, Chicago, October 17-20, 1994.

[2] G. Booch, Object-Oriented Analysis and Design with Application, The Benjamin/Commings Pub., Inc., 1994.

[3] J. Bunn, P. Palazzi, B. Rousseau, M. Smith, A Step Towards LIGHT - LIfe cycle Global HyperText. *World Wide Web and beyond in Physics Research and Applications*, San Miniato, Italy, March 14-18,1994. To Appear in *International Journal of Modern Physics*.

[4] D. Casper, ALEPH 101 - An Introduction to the ALEPH Offline System. CERN, ALEPH 93-26, SOFTWR 93-06.

[5] N. Drakos, The LaTeX to HTML Converter, World-Wide Web document, http://cbl.leeds.ac.uk/nikos/tex2html/tex2html.html

[6] C. Mazza et al., ESA Software Engineering Standards, ESA Board for Software Standardisation and Control, Prentice-Hall, 1994.

[7] B. Rousseau, M. Ruggier, Writing Documents for Paper and WWW. A Strategy based on FrameMaker and WebMaker. Proceedings of *First International Conference on the World Wide Web*, Geneva, May 25-27, 1994. Also in *Computer Networks and ISDN Systems* 27 (1994) 205-214.

[8] J. Rumbaugh, et al., Object-Oriented Modeling and Design, Prentice-Hall, Inc., 1991.