LIGHT/OO: an extension of LIGHT to OO development.

A.Aimar [a], L.M.Barone [a,1], P.Krysiuk [a], P.Palazzi [a], B.Rousseau [a]

[a] *CERN ECP/IPT*

The LIGHT (LIfecycle Global HyperText) system, developed at CERN in ECP/IPT, provides a solution to the problem of software documentation and maintenance in the context of High Energy Physics experiments (size, complexity, distribution of development teams). The new generation of HEP experiments (LHC, BaBar) is moving towards new software technologies, like OO design and use of C++. LIGHT is being extended to cope with these technologies, to handle more documentation formats (OMT diagrams, requirement documents) and to allow more efficient Web navigation, profiting from all the emerging Web technologies. We present the current status of LIGHT/OO and its future evolution.

Key words: WWW; OO Programming; Documentation;

1 Introduction

The LIGHT (LIfecycle Global HyperText) system[1], developed at CERN in ECP/IPT, addressed two problems: making all the documentation of large software projects available to all the participants through the WWW; and easing the maintenance of that software by automatically hyperising all implicit cross-references between all documents, including the code itself. The first major application of LIGHT was JULIA, the reconstruction program of the ALEPH experiment at CERN (`http://alephwww.cern.ch/LIGHT/`), later extended to other parts of the off-line production chain. LIGHT is currently in production in the ALEPH experiment.
The new generation of HEP experiments (ATLAS and CMS at LHC, BaBar at SLAC) is adopting more modern software technologies: FORTRAN is being

[1] On leave of absence from University “La Sapienza”, Rome, Italy

abandoned, Object Oriented (OO) design is used for modelling the data handling and reconstruction, software engineering discipline is widely accepted, and the preferred language is presently C++. At the same time the use and the technology of the WWW are now evolving rapidly, allowing more efficient navigation models through a Web of documents. Since the LIGHT architecture was conceived from the start to be open to any language and to any type of document, it is therefore natural to investigate its application to OO programming techniques. In this paper we describe a prototype of LIGHT/OO built to cross-hyperize C++ code, OMT diagrams and related documentation, done in close contact with the OO software group of the ATLAS experiment.

2 The ATLAS C++ Prototype

The ATLAS experiment at LHC is considering modern software technologies for its on-line and off-line software [2]. In particular, the OO group is designing OO reconstruction algorithms using StP (`http://www.aonix.com/Products/StP/stp.html`) as a development environment, and the OMT methodology [3]. We have built a mock-up in order to test various solutions (`http://www.cern.ch/Light/coolight/atlproject.html`) to hyperise and navigate C++ code, OMT diagrams and documentation using part of the available ATLAS C++ code. The mock-up showed that one could profitably use extensions of HTML, such as frames, to navigate the code and the related documentation. Following the conclusions of the mock-up, we have developed a prototype capable of building a Web for OO packages in C++. The components of this prototype follow the original LIGHT architecture, and are backward compatible. In the next section we discuss the details of this prototype.

3 Implementation Issues

The LIGHT/OO prototype implements the same architecture as the one in production for ALEPH [1], described in Fig.1. However improvements and extensions have been carried out on all the components of LIGHT. A new set of parsers has been implemented for the OO case along with a PATCHY parser which was missing in the first implementation. The PATCHY parser has been written to put under LIGHT software of DELPHI and OPAL. So the LIGHT "parser layer" now includes parsers for:

- FORTRAN77
- PATCHY
- ADAMO Data Definition Language

- C++
- C++ pre-processor directives
- OMT diagrams produced by OMTool
- OMT diagrams produced by StP
- C++ embedded comments (inline documentation)

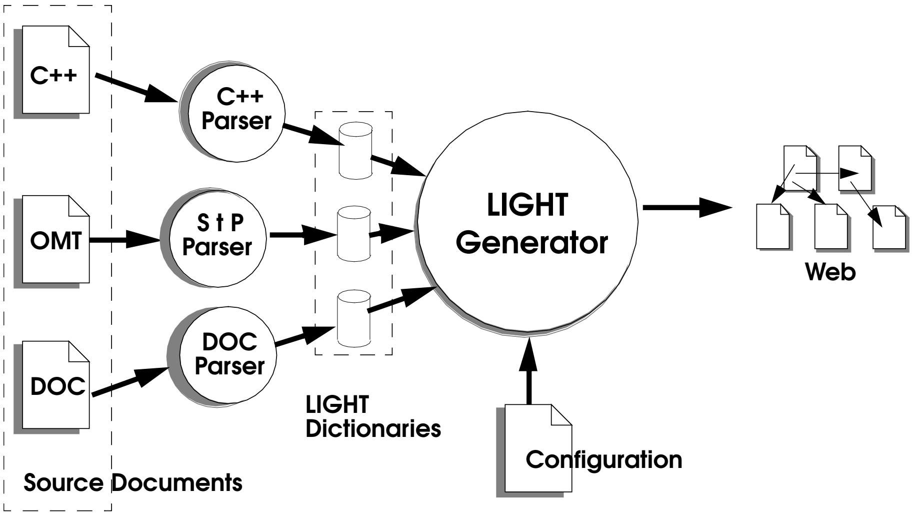

Fig. 1. The architecture of the LIGHT system.

The C++ parser is a home-made product, based on the PCCTS (`ftp://ftp.parr-research.com/pub/pccts/`) grammar; it currently has limited functionalities, especially in the area of recognition of methods.
The LIGHT Dictionary is based on ADAMO (`http://www.cern.ch/Adamo`), and contains all potential sources and targets for hypertext links (tokens). In the Dictionary the concepts of token scope and FileSet have been introduced, which were missing in the previous implementation. The token scope allows to recognize tokens inside nested scopes (e.g. local variables inside nested C++ classes). The FileSet makes the configuration of the Web more flexible. It is possible to process groups of files at a time with a different configuration for each group.
The LIGHT generator has been completely rewritten. It is now faster and more efficient; it generates HTML pages combining (parts of) input files, HTML template files, interacting with the dictionary for token recognition and link resolution. The generator is coupled to a completely redesigned configuration file, where the connectivity rules for various token types are specified. An example of configuration file for C++ is shown below, with the connectivity rules for some class-related token types:

```
{link:
local "class_usage" to "class_def" token
local "class_fwd_decl" to "class_def" token
local "class_inher" to "class_def" token }
```

The first rule links, for instance, the name of a class from which an object is created (`"class_usage"`) to the header file where the same class is defined (`"class_def"`), and so on. Of course the configuration file can be modified by the user, allowing a change in the connectivity rules. Finally, in order for LIGHT to generate a full Web, a set of utility programs and scripts (the "glue") not shown in Fig.1 is needed to generate indexes, to create "control" HTML pages, as framesets and navigation pages and so on.
The navigation model in LIGHT/OO is different from the one in ALEPH LIGHT. All the Web is built around few central entities (classes, methods, files); the use of HTML frames allows to browse an instance of these entities (e.g. a given class) through all its representations (the code, the documentation, the OMT diagrams where it appears). The user has then easier access to the complete description of the entity.

4 Future Developments and Conclusions

The LIGHT system appears to be an ideal solution to the problem of distributing software documentation and making software maintenance easier. We have shown the possibility to apply this solution to software projects using modern technologies such as OO methodologies and programming languages. Future developments of the project follow three main directions:

1) the application of LIGHT to free text documents (e.g. Help Guides, instruction manuals for on-line shifts);
2) an API to the Dictionary to ease interfacing new parsers to LIGHT, and to leave parser control to the users;
3) "on-the-fly" generation of HTML pages, either from the standard Dictionary, or from a more complete database.

References

[1] A. Aimar et al., Using WWW to Improve Software Development and Maintenance: Application of the LIGHT System to ALEPH programs, in: R. Shellard and T.D.Nguyen eds., *Proceedings of CHEP '95* (Rio de Janeiro, Brazil, 1995) 568–575.

[2] The ATLAS Collaboration, ATLAS Computing Technical Proposal, *CERN/LHCC/96-43*.

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