LASER INTERFEROMETER GRAVITATIONAL WAVE OBSERVATORY

- LIGO -

CALIFORNIA INSTITUTE OF TECHNOLOGY

MASSACHUSETTS INSTITUTE OF TECHNOLOGY

Technical Note **LIGO-T960052-00 - E** 6/3/96

LIGO Modeling Software Specification Plan

J. Kent Blackburn

Distribution of this draft:

Barry Barish, Gary Sanders, Robbie Vogt, Stan Whitcomb

Albert Lazzarini, Dennis Coyne, Hiro Yamamoto

Rai Weiss, David Shoemaker, Mike Zucker

Lisa Sievers, Dave Redding

Rolf Bork

Greg Hiscott

California Institute of Technology
LIGO Project - MS 51-33
Pasadena CA 91125
Phone (818) 395-2966
Fax (818) 304-9834
E-mail: info@ligo.caltech.edu
WWW: http://www.ligo.caltech.edu

TABLE OF CONTENTS

TABLE OF CONTENTS

1 INTRODUCTION

This LIGO Modeling Software Specification Plan (LMSSP) describes the software management and development process for the Modeling group within Systems Integration for LIGO. The techniques for software development and management outlined within this document are to be utilized project wide by individuals and groups developing modeling and simulation software for LIGO. Included within this scope is the successful integration of the modeling and simulation software with the data analysis software.

1.1. Project Overview

The primary purpose this LIGO Modeling Software Specification Plan is to guarantee that all software developed to LIGO modeling and simulation studies has a legacy. By adopting the methods outlined within this document software development for LIGO will have a future path for growth and integration with LIGO data analysis.

1.2. Project Deliverables

The project objective is to provide all modeling and simulation software and documentation necessary to LIGO. This includes noise models, IFO models, time domain models and the integration of these into the end-to-end modeling environment.

1.3. Evolution of the LMSSP

This is intended to be a living, working document over the lifecycle of the project. It will be reviewed for accuracy prior to any formal reviews, whenever higher level LIGO management policies are published to ensure adherence to LIGO standards, and when plan changes are approved which affect this document.

1.4. Reference Materials

1. AVS Software Standards Guide LIGO-T950057-00-E
2. AVS5 Documentation Set
3. AVS/Express Documentation Set

1.5. Definitions and Acronyms

AVS - Advanced Visualization System

AVS5 - Version 5 of the AVS user oriented modular software environment

AVS/Express - Developers' version of the AVS object oriented software environment

CVS - Concurrent Version System (Front-End to RCS)

IFO - Interferometer

LIGO - Laser Interferometer Gravitational Wave Observatory

RCS - Revision Control System

SCCS - Source Code Control System

TBD - To Be Determined

TCCS - Trivial Configuration Control System (Front-End to both RCS and SCCS)

VIZ/Express - Users' version of the AVS object oriented software environment (replaces AVS5)

1.6. Document Precedence

In the event of conflict between this document and other LIGO documentation, the order of precedence, for this particular project, shall be:

1. LIGO Science Requirements Document
2. This document
3. Reference 1

2 PROJECT ORGANIZATION

2.1. Process Model

The basic development process model is described in Section 4 of this document. Exact procedures and management processes will be in accordance with LIGO Systems Integration which oversees the modeling group. Project schedule and milestones are officially maintained by LIGO Systems Integration Management.

2.2. Organizational Structure

LIGO modeling and simulation software is to be developed as a collaborative effort between the Modeling Group under Systems Integration and the whole of the LIGO Project. The modeling group is made up of members of the Systems Integration Group. While the Modeling Group will have a primary role in the development of the software, since the final product will often have outside contributors and is intended for use by the whole LIGO Project and even the LIGO Research Community, all individuals involved share responsibility in the successful development.

3 MANAGEMENT PROCESS

3.1. Management Objectives

The primary goal of this project is to provide quality software which is an integral part of the

LIGO Project effort to provide fully functioning modeling and simulation tools which assist in the understanding and analysis requirements of LIGO. Management objectives toward meeting these goal are:

- Early guidance and planning of the project
- Collaboration with cognizant staff on physical and engineering models
- Testing and Verification of software models
- Risk Assessment and Analysis
- Establishing standard software procedures and coding areas
- Incorporating configuration control procedures.

3.2. Risk Management

Risk will be analyzed throughout the project lifecycle in terms of technical, cost and schedule risks. Risk analysis shall be presented at each review, along with mitigation techniques. Software standards will be used to the fullest extent possible to reduce longer term risk factors.

3.3. Monitoring and Controlling Mechanisms

Monitoring and control mechanisms shall be in accordance with LIGO project management plans.

3.3.1. Formal Reviews

Formal reviews of the modeling software will be held to evaluate design issues and to communicate the directions and options available for the integrated modeling software in accordance with recommendations from the System Integration group management and per the LIGO project management plans.

3.3.2. Informal Reviews

The modeling group software developers must work as an integral part of a team with other members of LIGO and the scientists assigned to the LIGO detectors to provide a tightly integrated, functional product. As part of this interaction, informal reviews will be conducted within the LIGO team of scientists and engineers.

3.3.3. Software Configuration Management

Software configuration management includes the activities of configuration identification, change control, status accounting, and audits. Baseline software configurations will be provided by the Modeling administrator to LIGO Modeling Configuration Control. Pre-baseline development configurations as well as baseline configuration will be controlled by the Modeling administrator.

3.3.3.1 Flow of Configuration Control

The software and documentation developed for the LIGO Modeling project will move through four distinct areas, as shown in Figure 1: Software Configuration Control Flow, to help maintain configuration control. The general flow of software and documentation is:

1. Development Area: Area in which software developers work on code in progress. This area has symbolic links and environment variables pointing to the Release Area, to ensure the developer is using the latest released versions of software operating systems and tools.

2. Archive Area: Once a developer is satisfied that particular code is ready for release, the code and documentation is moved into the Archive area using RCS commands. Here the code is archived along with necessary history and log records. Code may move back and forth between these first two areas as bugs/faults are detected and repaired, each time being assigned a new version number through RCS. Faults/desired corrections are documented with a Software Maintenance Request (SMR) (discussed later), which travels with the code and is maintained in the Archival area in the history records.

3. Release Area: This is the repository for all source code and documents which has passed test and is ready for build(installation). The assigned release engineer is then responsible for integration of all such code and coordinates the update into the Release Area. and the building of the software libraries, object files and executables in the Build area.

4. Build Area: This is where all installed LIGO Modeling software are located and made publicly available to users. Once the build area is populated with a new version, a suite of automated tests will be performed to verify reliability.

Further expansion and definition of these areas is covered in the Technical Process section of this document.

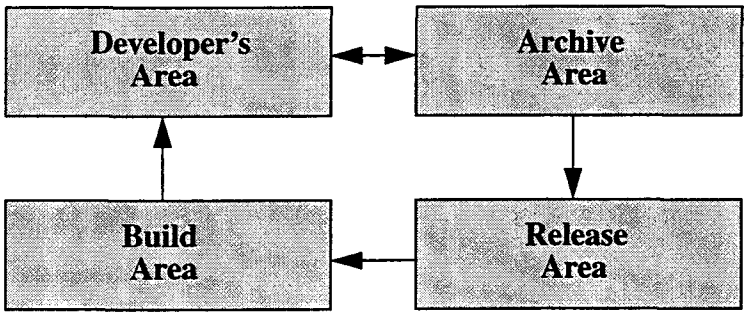

Figure 1: Software Configuration Control Flow

3.3.3.2 Configuration Control Tools

All software, in every area described above, will always be under source code control, using the Revision Control System (RCS). RCS is chosen over SCCS on technical merit and because

- it is a more modern utility for Source Code Management than SCCS

- it is more user friendly

- it is available freely from the FSF for both Unix and PCs

- it has several powerful Front Ends to choose from: TCCS, CVS

- a tools *sccs2rcs* is available to convert SCCS archive files to RCS archive files.

3.3.3.3 Configuration Identification

The configuration identification for each code module will be the revision number assigned automatically by the RCS. Once the Release Engineer has integrated the various code modules, and it has been approved by the Modeling Administrator for release, the integrated code will be put under a unified RCS revision number and released.

The release numbering scheme shall be a three number convention in the form x.x.x, such as 1.2.3. The first number shall indicate a major release. Major releases are typically limited to when the code has undergone major core structural changes or a significant number of enhancements have been made. The second number is changed when a release has new features/enhancements. The final number indicates that bug fixes have been made without the addition of particular features. Developer's may choose to implement a four number convention in their local development areas.

3.3.3.4 Handling of Project Media

All Modeling documents, source files and build instructions will be locally controlled by the Release Engineer. Whenever a Modeling product is approved and baselined (sent to the Release Area), a copy of all materials shall also be turned over to the LIGO Integration Group for LIGO Document Control.

3.3.3.5 Enhancements and Changes (Corrective Action)

3.3.3.5.1 User Service Request

Once software has left the development area, all requests for enhancements or corrections are documented in an USR. An USR has three basic parts:

1. Problem reporting/enhancement request area submitted by the software end user.

2. Analysis section, wherein the assigned software engineer analyzes the problem/request and provides recommendations to resolve the request.

3. Resolution Area: Information on how the request/problem was resolved.

3.3.3.5.2 Corrective Action Process

Once an USR is originated, it is submitted to the Modeling Administrator. It is then assigns both a priority and a developer to be responsible for analyzing/resolving the request. Priorities are assigned according to the following table.

Electronic submission methods of a TBD architecture will be utilized to guarantee automation, tracking and reliability. This electronic method could range from a simple archived mail exploder all the way to a database capable of generating reports depending on the cost and man-power available to implement the method.

Once the USR has been analyzed and response returned to the Modeling Administrator, it is

Table 1: USR Priority Assignment

Priority	*Description*
1	The problem adversely affects either an essential capability specified in the requirements and no work-around is known.
2	Same as 2 above, but a work-around is known which may be put in place as a temporary solution.
3	The problem causes inconvenience or annoyance but does not affect a requirement.
4	All others not falling into a category above.

reviewed and assigned for implementation. Here it undergoes the same procedures as apply for new software development. Upon completion of test, the USR is completed by the developer and returned to the Modeling Administrator for closeout.

From date of origin/receipt until closeout, the status of USR's will be updated on a weekly basis, with a status page made publicly available such that end users and management can be kept apprised of USR progress.

3.3.3.6 Configuration Management Documentation and Reporting

The primary reporting will be in the form of a Configuration Index (CI). During the design phase, software components will be identified, which are then tracked throughout the software lifecycle. One CI is prepared for each of these components. The CI includes an historical record section, milestone data, list of associated documentation, and a list of applicable USR or other project change requests (including status/disposition). The CI would reside under the same RCS system as the source code.

4 TECHNICAL PROCESS

4.1. Software Development Life Cycle

Software development will follow the basic standard modification life cycle as much as possible.

This cycle is shown in Figure 2: Modeling Software Development Cycle. While this outlines the general flow of development, reiteration between certain phases will occur, for instance, prototype and test may indicate that requirements need to be changed/updated or a new approach taken.

4.1.1. Developers

Developers create or modify source code based on design, enhancement and bug fixes. The *developer* will test and verified the performance of the source code in the *developer's* own local area. Once the source code has been verify the *developer* will archive the source code and all ancillary files such as Makefiles, Documentation, etc. through the source code management tools.

4.1.2. Source Control Archive

The source control archive will consist of the archives(files) located under the modeling directory structure. For each source code there will be a separate archive. The archive will contain all the components for revision control, access control, logging and history. These mechanisms will be used by the *developers* to maintain a clean archive. Source code archived will be readable by all members of LIGO but the archive can only be written to by the developer of the source code or the modeling *administrator.*

4.1.3. Release

The archived source code will not contain the needed libraries, object files and binary executables that are needed to use the code. However, all that is needed to construct usable code will be found in the archive. The *release engineer* will use the files contained in the source code archive to deliver a "release" of the software. This will involve extracting the "release" versions of the source code. These files will be used to modify or create the header, source code and documentation subdirectories for that particular release located under the modeling directory structure and to build in the build area the associated libraries, object files and binary executables. This will require that the *release engineer* have unrestricted access to the modeling directory structure.

4.1.3.1 Administration

The *administrator* of LIGO modeling project oversees the maintenance on the modeling directory structure and acts as the mediator between the develops, release engineer and users of the software to coordinate access and availability of files controlled by the LIGO modeling project. The *administrator* will have unrestricted access to the modeling directory structure and will be capable of preforming the release process. The *administrator* will also be able to write to all source control archives. The *release engineer* and the *administrator* are often one in the same.

4.1.3.2 Users

Users of the modeling software must have read access to the files needed to use the software. The *user* will also play an important role in the enhancement and debugging of software in the release area. The *user* reports bugs and requests for enhancements through the User Service Request mechanism.

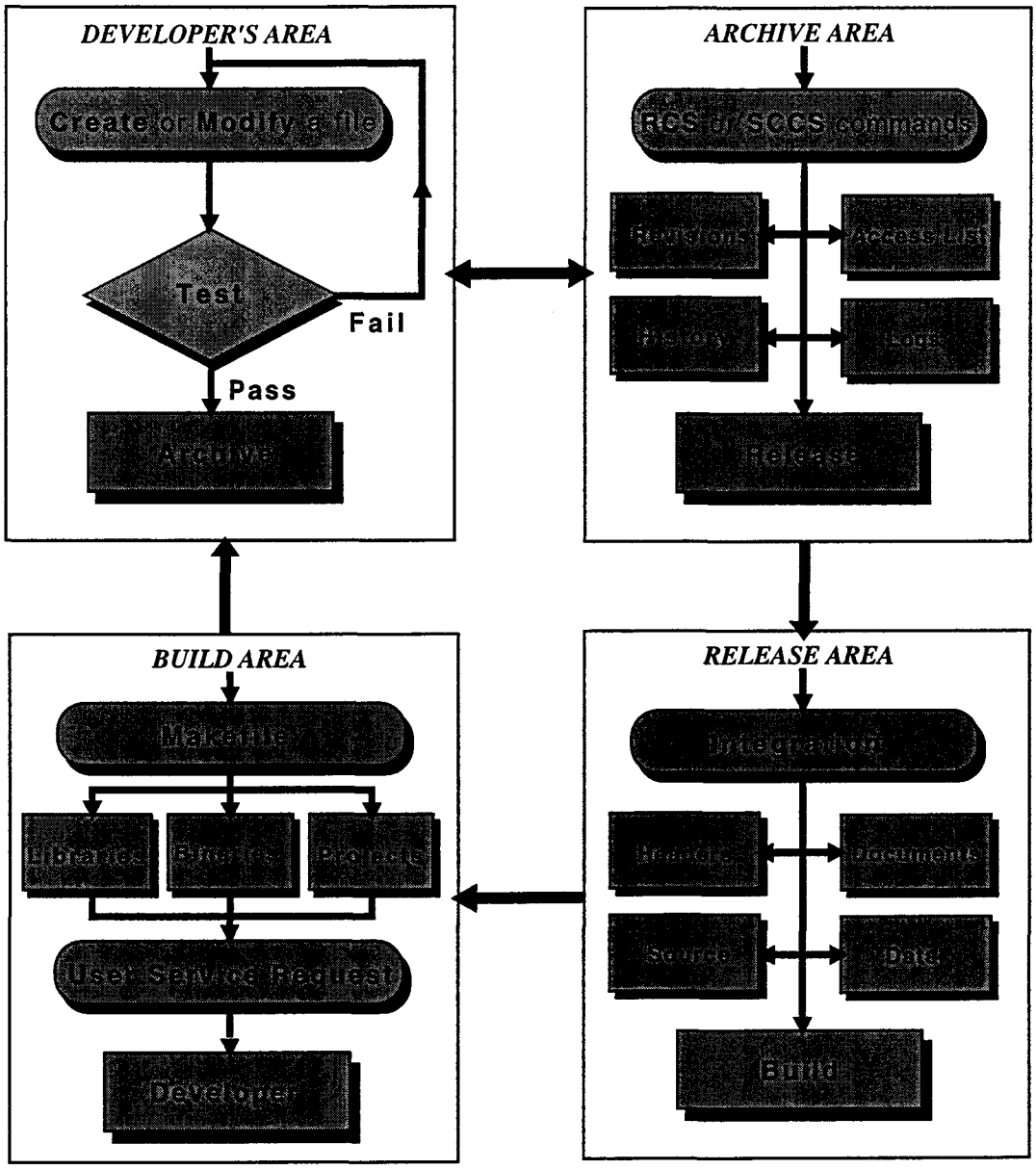

Figure 2: Modeling Software Development Cycle

4.2. Software Development Methodology

4.2.1. Software Standards

This software will be developed in accordance with the standards of the LIGO Modeling group for all software development projects involving modeling and simulation. These standards are given in the proceeding sections.

4.2.2. Design and Development Tools and Techniques

4.2.2.1 AVS and Development Tools

The LIGO modeling software will be designed and developed based on the present capabilities of the Advanced Visualization System (AVS) software package,. For this project, AVS will be used to design and develop requiring the use of both the VIZ[1] and the Express products. Both of these packages are high end data visualization products based on the object oriented paradigm.

AVS will provide a common environment for all LIGO software. This includes the modeling software, data analysis software and user supplied software. The advantage this brings to LIGO software is the standardization of the interface between software components, allowing for example the use of modeling software in the data analysis software

4.2.2.2 Programming Languages

The programming languages to be used are driven by compatibility with the AVS software environment. This environment supports C, C++, and FORTRAN[2]. Mathematica and Matlab are also available indirectly through layering with C and FORTRAN. In the case of C and FORTRAN the ANSI standards will be used whenever possible. Special cases where non-ANSI extensions to a language are necessary to take full advantage of a particular hardware platform such as the case with code developed for the parallel computers shall be considered within the specifications.

4.2.2.3 Documentation Tools

All software will be documentation at three levels to provide both programmers and users of the software:

- Comment statements will be used to document code

- Source Code Management Tools will be used to document version and history of code

- On-line documentation "help files" will be integrated into the software user interfaces

- LIGO standard publishing practices will be used to generate integral modeling and simulation software package documentation.

4.2.3. Target Systems

Target systems are defined as those hardware platforms and software systems on top of which the LIGO modeling software will operate. In general, these systems are constrained to those that support Unix and the AVS environment. Special application computers such as the INTEL Paragon, a parallel computer, will be used as needed to enhance the needs of the project.

---

1. VIZ is the user version of Express and will be available in the spring of 96
2. FORTRAN with be available in the AVS/Express 3.0 (expected shipping data mid 96)

4.2.3.1 Operating Systems

The operating system for LIGO modeling software will be Unix. The only restriction beyond this is that the development tools employed by this project be available and integrated for these operating systems.

4.2.3.2 Hardware Resources

The principal software development hardware will be Sun workstations. However, other hardware platforms will be used to assist in design and development for software that will be ported to the Unix environment when deemed appropriate. Special software requirements that need high performance computer hardware will be developed on the parallel computers available to LIGO.

Figure 3: Modeling Subnet Topology shows the modeling subnet of LIGO's local network at Caltech which is to be established for the modeling effort. This subnet will consist of high performance Sun workstations such as the Sparc20 using a common compute/file server in isolation from the other sub-nets of LIGO via a LIGO router. This will optimize the performance of these computers and provide the needed bandwidth for distributed computing.

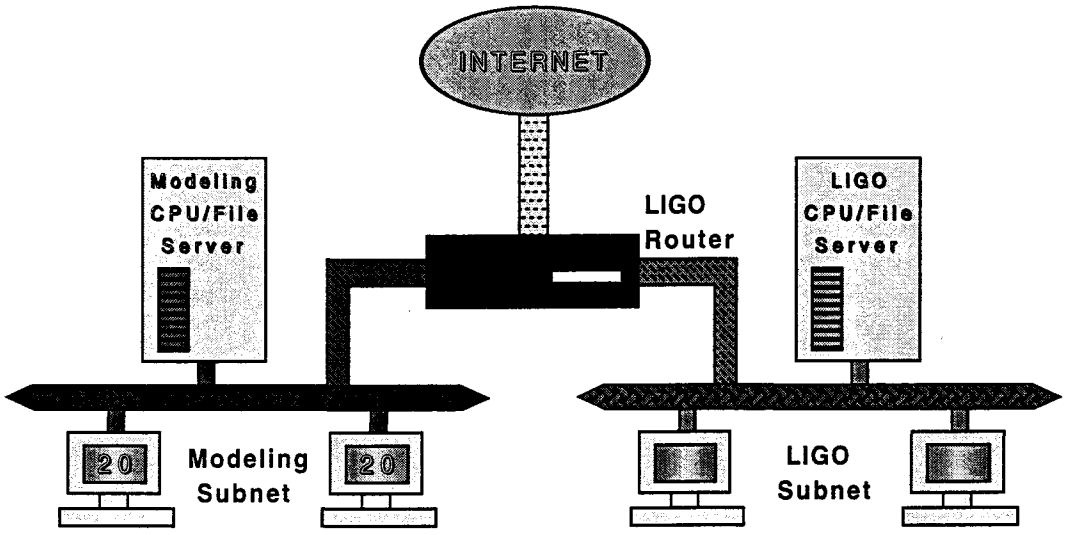

Figure 3: Modeling Subnet Topology

4.2.4. Security

The security of modeling software will be handed by the POSIX standard file protection scheme as it is implemented by standard source code management guidelines.

4.2.5. LIGO Modeling Code Directory Structure

Figure 4: LIGO Modeling Directory Structure shows the planned layout of the LIGO modeling software area. At the highest level will be a modeling home directory located on the modeling cpu/file server. This directory will be exported to all the workstations on the modeling subnet., Below this directory are the individual subdirectory for managing the source code and making the software available to users. All directories, subdirectories and files will be read/write accessible to

the release engineer and the administrator. The developers will have read and write access to the archive and its subdirectories. The general user will have read access to all levels of this directory structure.

4.2.5.1 Archive Directory

The archive directory will contain individual subdirectories for each software package developed. The specifics of these subdirectories are determined by the choice of source code management utility. The basic contents of these subdirectories are the revision archive, history file, log file and access name file. This directory will be readable to the world and writable to the developers and the release engineer.

4.2.5.2 Build Directory

The build directory is disk space used during the build of the releases. Files created in this area are temporary and shall be removed at the completion of the build procedure. This directory will be owned by the release engineer and read only to all others.

4.2.5.3 Integration Directory

The integration directory contains the subdirectories associated with each release of software and project. Each subdirectory contains the scripts/utilities used to bind together the source revisions located in the archive. This directory will be owned by the release engineer and read only to all others.

4.2.5.4 Bin Directory

The bin directory contains the executables for the current release of all software developed by the LIGO Modeling Group. files in this directory will be owned by the release engineer and read/execute only to the general user.

4.2.5.5 Include Directory

The include directory contains the header files used to build the current release of all software currently distributed via the bin directory. Files in this directory will be read only to the general user.

4.2.5.6 Lib Directory

The lib directory contains the current release libraries available for development and used to build release software. Library names will follow the Unix convention of libxxx.a as part of the LIGO Modeling software standard. Files in this directory will be read only to the general user.

4.2.5.7 Man Directory

The man directory contains Unix man pages for Unix-like software developed by the LIGO Modeling Group. Files in this directory will be read only to the general user.

4.2.5.8 Docs Directory

The docs directory contains subdirectories for help, frame, html, man#, ps. The help subdirectory contains text based files that are access by standard unix editors and the AVS on-line help interface. The frame subdirectory contains framemaker file with names that follow the convention

xxx.fm# where the # identifies the version of framemaker needed to view the file. Files in this directory will be read only to the general user.

4.2.5.9 Projects Directory

The project directory will contain AVS bundled files. This includes the "networks" files of AVS5 and the projects files of AVS/Express and VIZ. Subdirectories for each network and project will be located under this directory. The directory and its files will be owned by the release engineer and readable by all users.

4.2.5.10 Data Directory

The data directory contains the data file used by all LIGO Modeling software. Any software developed which requires data files shall be developed to look in this directory at a minimum as it searches for needed files.

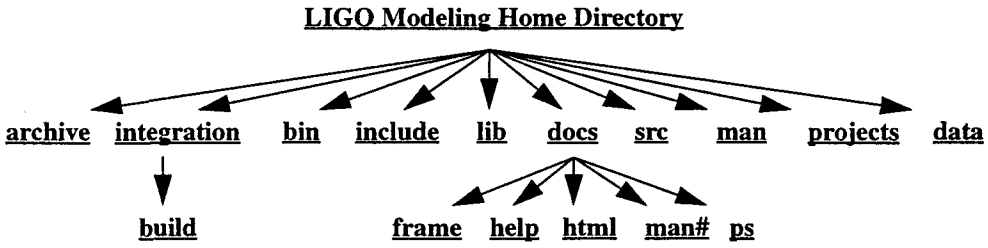

Figure 4: LIGO Modeling Directory Structure