THE 17th CHESAPEAKE SAILING YACHT SYMPOSIUM

ANNAPOLIS, MARYLAND, MARCH 2005

The Development of an Integrated Ship Design Environment for the Naval Architect on the Linux Operating System

H. James Parker, Gibbs and Cox, Inc.

ABSTRACT

When naval architects design a new yacht or ship, they must use many separate programs to design, analyze, and create the production drawings. These separate programs are most likely from different vendors using non-compatible data formats. The translation process between all these data formats can take many non-productive hours. In addition, any changes to the design often affect many separate drawings and software programs that must be changed independently

GtkCAD is a new program that attempts to solve many of these problems by integrating these many individual programs into a program suite with a single user interface and a common data format.

GtkCAD's modular design allows the user to load plug-in modules at runtime to add the functionality needed for the task at hand. The modular design also allows the user to customize GtkCAD to suit their particular needs at any given time. For example the hydrostatics program, used by naval architects, may not be very useful to civil engineers. This potential cost savings feature allows a customer to custom-build a system based upon their needs.

Linux is a true multi-user Operating System, and GtkCAD is designed to take advantage of this by allowing multiple engineers to use a single binary copy of the program simultaneously. The use of shared libraries lowers the memory footprint by allowing multiple users of the program to share a single copy of the library, and also saves time during software upgrades. The use of a SQL database also allows controlled access of the data by several engineers simultaneously, remotely if necessary.

GtkCAD is being developed using the Open Source development model. With this model groups of diverse software developers work on a common software project, often volunteering their time. Anybody is allowed to contribute to the project. In return, the software is freely distributed to the end user with little restrictions on its use, modification, or redistribution.

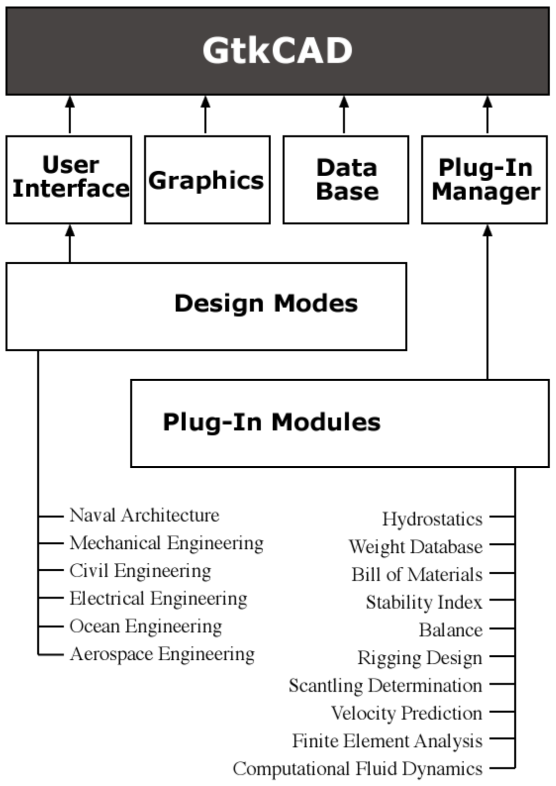


Figure 1 GtkCAD Program layout

NOTATION

API	Application Programming Interface
BOM	Bill of Materials
BRep	Boundary Representation Solid Model
CAD	Computer Aided Design
CAE	Computer Aided Engineering
CAM	Computer Aided Manufacturing
CFD	Computational Fluid Dynamics
FEA	Finite Element Analysis

FELT	Finite ELemenT analysis program
GIMP	GNU Image Manipulation Program
GNU	Free Software Foundation's Software System
GUI	Graphical User Interface
IDE	Integrated Design Environment
ISO	International Standards Organization
SQL	Structured Query Language.
STIX	ISO Stability Index

OVERVIEW

This paper describes the design and development of GtkCAD. GtkCAD is a free, modular Computer Aided Engineering (CAE) design environment for the Naval Architect. GtkCAD takes advantage of Open Source development methods to create a robust and modular naval architecture design system on the Linux operating system.

The goal of GtkCAD is the development of a software system that allows the design of a yacht, from requirements to manufacturing, using the design spiral methodology, with a single user interface and a common data format.

Most of the computer work created by naval architects is accomplished using many unrelated programs. A significant amount of time is spent transferring the data between the different programs. By using a common data format, GtkCAD has the potential to save many man-hours in the design process by eliminating the data transfer process.

Another problem facing Naval Architects today is the use of many related but unconnected drawings to convey important information. On a large project, such as a Ron Holland's recent super yacht, ***Mirabella V***, a single simple change may affect hundreds of drawings and computer models, all of which need to be changed one-by-one. By using a common database and data format for all project data, GtkCAD will allow the change to propagate to all the project drawings either automatically or by simply rerunning the appropriate plug-in module.

One of the advantages of using the computer to solve tedious engineering calculations is that it frees the designer to explore multiple ways to solve the problem, and optimize the solution. GtkCAD offers the designer a chance to do parametric studies of potential solutions very early in the design spiral. By using a plug-in architecture, GtkCAD allows unlimited flexibility in the type of analysis that may be performed.

The complete software system will consist of a three dimensional modeler, where the initial competing models are developed. The designer will then be able to use dynamically loaded plug-in modules, or plug-ins, to analyze the design, and choose among the competing models. Once a model is chosen, the designer will then be able to create two dimensional construction drawings, and develop CNC tool paths, for use by the boat builders.

During the design phase, the system will automatically create a bill of materials (BOM) and maintain a database to keep track of weights and positions, and costs of material pieces, which are critical to successful designs.

Writing reports are another central activity in any engineer’s career. Most of GtkCAD's modules provide the functionality to automatically plot the results of the analysis in a form ready to be included in design reports. This is another potential time saving function, as it will no longer be required to transfer the data to another software program to create tables and charts.

The main program is a generic three-dimensional modeler. GtkCAD was designed on a generic basis, so that many diverse engineering disciplines can take advantage of the common source code base. GtkCAD's interface can be modified at start-up by loading special modules, called design modes, specific to an engineering discipline. The current release of GtkCAD should be considered a framework for future development.

THE DESIGN SPIRAL

Naval architecture uses an iterative approach in the design of boats and ships. With each iteration the design evolves further into its final form. During this design evolution many variations must be created and evaluated, so as to optimize the final design, and resolve the many conflicting requirements.

These steps are often referred to as the 'Design Spiral'. The requirements phase is the entry point, and each of the above stages can be thought of as a separate loop of the spiral as it converges to the final design at the center [Hollister][Larsson].

Each loop of the spiral can be further divided into segments. Each segment can be thought of as a particular aspect of the design, such as hydrostatics and stability or performance evaluation. Not all aspects of the design need to be evaluated during each design iteration [Larsson].

Yacht design usually involves many conflicting requirements. This causes the designer to create solutions, and evaluate the many trade-offs necessary to optimize the design. As is the case in any engineering discipline, the earlier in the process the trade-offs are resolved, the cheaper the design process becomes. A mistake found in the detailed design or production may require an entire rework of the design [Hollister].

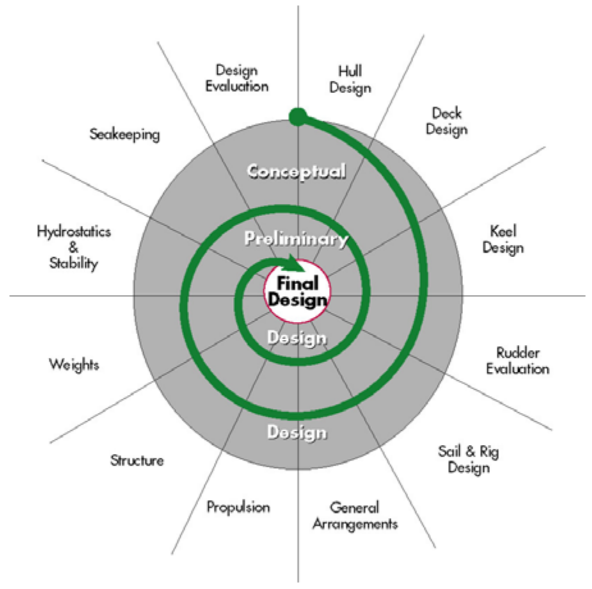


Figure 2 The Naval Architecture Design Spiral

GtkCAD is designed to be an integral part of the design spiral process. Using its modular approach, the designer can build multiple competing models, and analyze those quickly and easily using easily loaded modules, without the need for data conversions between incompatible programs. As the design progresses through the spiral, the analysis can become progressively more detailed by using the appropriate plug-ins. This will all be accomplished without ever leaving the GtkCAD interface.

Requirements: In this stage, the potential design is discussed with the client and develops a statement of purpose and a prioritized list of desired features developed. Stephan Hollister also suggests the development of a 'measure of merit' formula to quantitatively evaluate various design trade-offs [Hollister].

The requirements stage of the design spiral is usually accomplished by meetings the client(s) and discussing the upcoming project. The end of this phase is a document, that when approved by the client, will guide the design process.

The requirements phase is not easily incorporated into the design of GtkCAD. Other than data storage of the requirements document, and its revisions, there is not much to automate in this step.

One of several things that can be accomplished at this point is the development of the measure of merit. A measure of merit is usually an algebraic formula where each desired feature of the design are weighted. The designer can then run the formula through a special program called an optimizer, to find the optimal solution to what normally are many conflicting requirements.

GtkCAD will eventually include a scripting language, which could be used to develop an optimization routine based on a measure of merit. Another option would be to provide a spreadsheet type widget to show the relative merits of each model in tabular and/or graphic forms.

During the requirements stage, a designer may also complete a market research study to determine how the problem may have been solved in the past. GtkCAD includes a small database to store the published characteristics of boat designs. A simple analysis program will assist the designer in deciding the most likely hull form parameters and proportions.

Conceptual Design: During this phase, a determination is made if it will even be possible to meet the requirements, or if the requirements will have to modified. Several different concept designs may be produced for preliminary evaluation. Preliminary 3D models are created and optimization programs are used. Usually a design proposal is written at the end of this phase [Hollister].

During this phase the designer must identify all design tradeoffs that will be influence the design. This is where the measure of merit, developed in the requirements stage comes into play. With a properly designed measure of merit, the designer can fully understand the tradeoffs that will be required and how they affect the design.

From the above trade studies, the designer will then be able to create several candidate designs for evaluation. These designs may come from the imagination of the designer, or synthesized using a software program, such as NAVSEA's ASSET program. GtkCAD in future releases should be able to provide such a synthesis tool. Once the competing designs are developed, they are then analyzed and compared using the appropriate evaluation tools. These tools are the strong point of using a program such as GtkCAD. GtkCAD's plug-in architecture allows the naval architect the ability to load the appropriate tool into the program, and compare several models in parametric studies.

Preliminary Design: In this phase one of the proposed conceptual designs is chosen as being the best to meet the requirements and the design is finalized and evaluated more fully. For example CFD and FEA calculations would be performed during this phase. The final 3D model is also completed during this phase.

The preliminary design is where a program like GtkCAD can prove its effectiveness. The plug-in architecture will allow the naval architect to analyze the design, make adjustments, and rerun the analysis without needing to change programs or worry that the changes will propagate throughout the design. If a history is kept, then the evolution of the design is maintained, and lessons learned recorded.

During this stage a detailed bill of materials is started for accurate weight estimation and cost analysis.

Detailed Design: Often called the final design, it is during this phase the 3D model is used to create 2D construction drawings for use by the builder. It is also during this phase that all CAM programs are written and verified.

It is during this stage that GtkCAD will likely be used to develop the detailed 2-D construction drawings from the 3-D models, using the CAD plug-in module. The CAM plug-in may be used to develop and test the tool paths used to create the molds and cut the bulkheads and frames. The Bill of Materials would be finalized so orders for the necessary equipment can be placed well ahead of the time they are needed during construction.

EXAMPLES OF CURRENT PLUG-IN MODULES

Active development of GtkCAD has mainly concentrated mainly on the development of plug-in modules. The goal of GtkCAD has been the creation of a complete suite of programs for the design of sailing yachts.

The following plug-ins are under active development:

Market Database: A simple database of the published data of boats built in recent years. By selecting some key parameters, a database query will retrieve boats with similar parameters, plus will compute the average and standard deviation of all the major parameters. This information can then be used as a starting point on the new design.

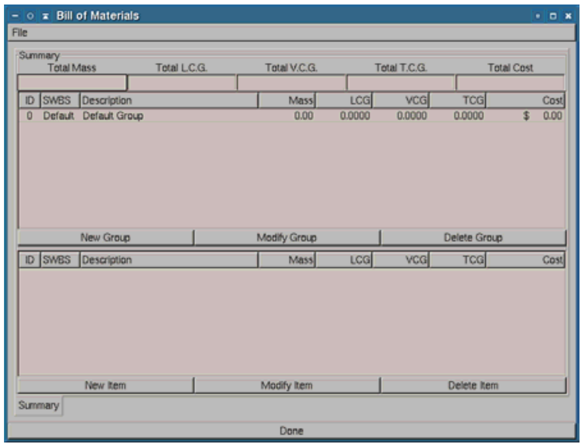
Hydrostatics: A simple hydrostatics module was created for GtkCAD. It has several purposes. First is to demonstrate how a plug-in can interact with the model and database. Second is to provide a native plug-in for the Naval Architect to calculate hydrostatic properties of the current model.

The hydrostatic performs its calculations using a table of offsets. It performs the necessary numerical integration using Simpson's first-third rule. While this is reasonable accurate, the creation of the table of offsets is inefficient and may be inaccurate.

The hydrostatic module is in its initial stages for intact stability of monohull vessels. It lacks the code for longitudinal trim, and weight addition and subtraction. However, it has the ability to graph the results of the calculations, although currently it only graphs section areas for the designed waterline, for demonstration purposes.

Future development of the hydrostatic module should include damage stability, tanks with free surface, and weight addition and subtraction. These additions will need to wait until the code for internal subdivisions becomes more mature. Another addition that will need to be included in any addition to the code base is the inclusion of appendages in the calculations. Finally the code must be expanded to properly solve the hydrostatics for multihull yachts.

Weight Database and Bill of Materials: The correct location of the center of gravity is an essential part of a successful design in Naval Architecture. Not only is it important for the trim of the vessel, but the stability, and hence, safety of the crew and vessel is dependent on it.



Total Mass	Total L.C.G.	Total V.C.G.	Total T.C.G.	Total Cost

ID	SWBS	Description	Mass	LCG	VCG	TCG	Cost
0	Default	Default Group	0.00	0.0000	0.0000	0.0000	$ 0.00

ID	SWBS	Description	Mass	LCG	VCG	TCG	Cost

Figure 3 Bill of Materials Dialog Window

To obtain an accurate location of the center of gravity, naval architects must keep track of every piece of equipment and structural member that is being used in the vessel, down to the last nut and bolt. This may be done using either a simple spreadsheet, or using expensive custom software. However, it is usually not part of the design software, so extra man hours must be devoted to this critical task. On very large projects, this might be a full time job for a weight engineer.

Another problem faced by designers is the creation of a Bill of Materials, for use during the construction phase, purchasing, and outfitting.

GtkCAD tries to eliminate both of these problems by integrating a Bill of Materials module. This module will automatically keep track of the weights and locations of equipment and structural members as they are added to the design.

Stability Index (STIX): Part of the new ISO standards, included the creation of a stability index to classify the stability of yachts. The stability index takes the most important aspects of seakeeping and stability, and calculates factors. These factors are then combined algebraically to from the STIX number [Larsson].

GtkCAD has included a module to calculate the stability index. The STIX number is also used by the Scantlings module to determine the proper scantlings for the vessel.

Balance: A simple module was developed to estimate the balance of a sailing yacht based on the method presented by Dr. Gritsmma, and modified by Larsson in "Principals of Yacht Design" [Larsson].

Rigging Design: GtkCAD's rigging plug-in module is based on the Nordic Boat Standard (NBS) as published in "Principals of Yacht Design". These rules were chosen because it is on of the few scantling rules that take the rig into account [Larsson].

The Nordic Boat Standard is useful because it covers both masthead and fractional rigs, with one or two pair of spreaders. Additional spreaders or unusual rigs need to be more fully engineered using finite element analysis, or other standard engineering practices[Larsson].

Scantling Determination: With the creation of the European Union as a political entity, a new common set of standards are being developed for products sold throughout Western Europe. The recreational pleasure boat market is no exception. The International Standards Organization (ISO), is currently (or has just finished) the new standards for the stability criteria and scantling determination of small craft less than 24 meters. The new standards cover the common building materials of wood, aluminum, steel, and Fiber Reinforced Plastics (FRP) or Glass Reinforced Plastics (GRP), or commonly called Fiberglass.

GtkCAD has included a module to calculate the scantlings using the new ISO standards.

Velocity Prediction Program: A Velocity Prediction Program is a specialized program used in sailing yacht design to predict the speed of a sailing yacht under varying wind speeds and directions. It solves the equilibrium equations affecting a sailing yacht to predict the yachts equilibrium condition.

A sailing yacht obtains a steady-state motion by balancing a series of conditions in equilibrium [Larsson].

1) The sail drive force must be balanced by the total resistance.
2) The sail side force must be balanced with the underbody hull, keel and rudder side forces.
3) The buoyancy force is equal to the gravity force. It is assumed that the vertical components of sail forces cancel out the vertical components of keel forces.
4) The heeling moment must be balanced with the righting moment.
5) Pitching moment of hull is balanced by restoring moment of the hull.
6) The total yawing moment is zero.

Velocity Prediction Programs use an iterative approach to solve the equations of motion.

GtkCAD's VPP plug-in module uses the calm water hull resistance as predicted using published model series data from the Delft University in the Netherlands, as originally published by Dr. Hazen and reproduced in the "Principals of Yacht Design."

Dr. Hazen's resistance calculation is based on a regression analysis of model tests, assuming a resistance is a function of length-beam ratio, canoe-body draft, displacement, longitudinal center of buoyancy, prismatic coefficient, and displacement-length ratio and Froude number [Larsson].

The sail force prediction also uses the published results of Dr. Hazen. These are based on the nominal sail area and lifting line theory. The model provides coefficients of lift for the five major sails for five separate wind angles from 27 degrees to 180 degrees. This method takes into account any blanketing or interference effects from sail trim. Additional data points may be determined using spline interpolation [Larsson].

The moment due to the side force from the sails must be counter-acted by the righting moment of the hull. When the two opposing forces are equal, the vessel will maintain a steady heel. These moments are taken through the longitudinal axis of rotation, assumed to be the waterline.

The righting moment is obtained from the hydrostatic module described above. The side force due the sails is calculated via the equations presented by Dr. Hazen for sail forces. The lift and drag forces are then resolved to obtain the side and driving force. The side force is further resolved to take into account the effects of heel [Larsson].

Finite Element Analysis: While designing GtkCAD, it was decided to use an external FEA code base. This had several advantages. First, a general purpose FEA package takes many man-years to develop and optimize, so it was not possible to consider such an undertaking. Second, it eliminates the duplication of coding efforts by multiple development teams. It is far more productive to use a synergy between teams, and concentrate development efforts at the team's core competencies.

FELT (http://felt.sourceforge.net) is an elementary Finite Element Program written to teach the Finite Element method to university engineering students. Because of its design origins, it cannot solve the full range of analysis problems that some commercial packages can. It also does not have the full range of elements in its library some of these competing programs have. The advantage of this project is that has been around for a number of years, so it has a large user base and a stable code base. Additionally it was designed in such a way as to easily expand the existing library as the need arises. It was also felt that the problems the software did solve were similar to the ones needed by the naval architects, thus eliminating the objection about not being able to solve a full range of problems.

Because FELT was not written as a native GtkCAD plug-in, it is not loaded and run in GtkCAD like a native plug-in would be. When the FELT plug-in is called a small program acts like a translator between the two programs.

FELT works by reading an ASCII text file that describes the problem and sets up the initial conditions. The GtkCAD plug-in works by writing this file for the current model. The module then starts FELT via a system call and monitors FELT's runtime status.

Computational Fluid Dynamics: CFD has been used with great success, where the competitive advantages overcome the high budget constraints. For example, all the major America's Cup teams use CFD, often in association with cutting-edge university research teams. As CFD techniques improve, it is expected that it will eventually replace the current tank-towing tests of models currently in use.

Gerris (http://gfs.sourceforge.net) was chosen due to its favorable license, its programming language, and the technological innovations it contains. The program has been downloaded and installed on the author's computer. Initial runs of simple two dimensional problems have yielded promising results.

One of the advantages of using Gerris is that it uses an adaptive mesh technique during its computational runs. In other words, Gerris adopts the mesh describing the fluid to the changing conditions of the flow, using a dense grid in areas of interesting flow, and using a coarse grid in other areas. This allows Gerris to capture vortex shedding around foil tips, while using relatively small computational resources [Popinet].

Gerris works by reading an ASCII text file that describes the problem and sets up the initial conditions. The GtkCAD plug-in will work by writing this file using initial and boundary conditions obtained via dialog boxes presented to the end-user. The module then starts Gerris via a system call and monitors Gerris's runtime status.

Gerris's models are described using tessellated polygons. This is defined to mean that the surface is divided into many small flat polygon surfaces that are used to approximate the actual surface. In Gerris's case, the models are described using the GNU Triangulated Surface Library. A second plug-in module is used to write the description file of the models geometry.

The author of Gerris has also written a simple program for post-processing and reviewing the results of the computations. It is hoped that this program can be adapted for use as a GtkCAD plug-in.

Suggested Future Plug-ins:

To avoid duplicating the efforts of other software development teams, whenever possible, GtkCAD uses the software developed by other projects. The majority of this code is used as plug-in modules to extend the functionality of GtkCAD. Current examples of these design synergy is the CFD and FEA modules described above.

The following Open Source projects are suggested as possible plug-in modules:

xFoil (http://raphael.mit.edu/xfoil): While designed primarily for aeroplane wings, it has been used with great success for the design of rudders and keels.

LinuxCNC (http://www.linuxcnc.org): This is a library to provide an interface to CNC machine control software. GtkCAD would provide an ideal CAM interface.

There are of course many more Open Source projects that might be adopted for use with GtkCAD. GtkCAD hopefully has been designed to be as compatible as possible with other projects.

There is also the possibility of using proprietary programs with GtkCAD. All that is required is that the program have documented and published methods of interacting with the program, such as ASCII file program inputs. For example, included with the Naval Architecture design mode module are import and export routines for the proprietary programs GHS (General Hydrostatics Software) and SMP91 (Ship Motions Program).

OPEN SOURCE DEVELOPMENT

One of the more interesting developments in software development that has occurred in recent years is the rise of the Open Source development model. With this model groups of diverse software developers work on a common software project, often volunteering their time. Anybody is allowed to contribute to the project. In return, the software is freely distributed to the end user with little restrictions on its use, modification, or redistribution. The most common licenses only require that any modifications and improvements to the software be given back to the development community, when the modified versions are distributed externally.

These licenses are usually based on existing Copyright law, and copyrights remain with the original author of the contribution.

It has been the intention from the beginning to release GtkCAD's core program modules to the public under the Free Software Foundation's General Public License, (See http://www.gnu.org). It is felt that this will allow the program to be widely distributed and a large user community to develop relatively quickly. The plug-in modules will be released under whatever license the copyright holder feels is appropriate. This is to encourage authors of proprietary software to develop plug-ins for GtkCAD without fear of Intellectual Property theft.

The modules that were developed for GtkCAD will be released under the Free Software Foundation's Lesser General Public License (See http://www.gnu.org), unless restricted by some other consideration, such as Trade Secrets of the author.

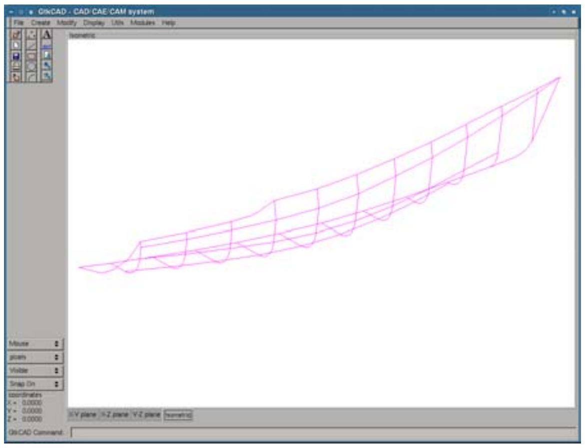


Figure 4 GtkCAD Main Interface Window - Isometric View

As with all large projects, GtkCAD depends on the active involvement of all participants for it to develop and grow. Thus part of any effort to promote Open Source software must involve the development of a community of users and the recruitment of software developers.

One important thing to remember is that just because you can not write software, does not mean you can not participate in the community. One of the most important things that needs to be done is to suggest improvements to the program, and report bugs when things don't work as they are supposed to.

For those who may want to contribute to the development efforts of GtkCAD, the following sections give a very brief description of the internal workings of GtkCAD.

MODULES

To keep the footprint of the software as small as possible, while at the same time giving the most functionality, GtkCAD makes extensive use of dynamically loaded plug-in modules, or plug-ins. Plug-ins are separately-compiled shared libraries that are loaded into the program at run time. This offers two advantages. The first is that the core program remains small and uses few resources. The second is that the program is very flexible. If the user does not need a feature, it will not be loaded into the program, making it possible for a designer to buy only the modules needed for a given activity, and thus save money.

The core GtkCAD program is a three dimensional modeler, which consists of three modules as explained above. In theory each module can be substituted by a module of equivalent functionality if desired. The first module is the user interface (UI) also commonly called a Graphical User Interface (GUI). The second is the graphics library, which draws the models on the screen. The third is the database library, which stores, retrieves, and modifies the model data in the database.

To make the three-dimensional modeler more useful to specific engineering disciplines, GtkCAD makes use of special plug-in modules called Design Modes. Design Modes are loaded at program startup and add functionality for specific engineering needs. For example, in Naval Architecture mode, when the user starts a new design, the design mode loads the New Boat dialog.

On top of these core libraries will be the plug-in modules, which extend the functionality of GtkCAD. Some examples of these plug-ins might be dimensioning, printing, and computer-aided machining (CAM). For the naval architect, modules such as hydrostatics, velocity prediction, resistance, computational fluid dynamics (CFD), and structural analysis are in development, as discussed above.

Another advantage of using plug-in modules is that one does not have to recompile the entire program to create and add a new module. Once a new program is created, the user just copies the plug-in to a specified location, and registers the plug-in with GtkCAD. From then on, the plug-in is available for use by the designers.

Types of modules

Three different types of plug-in modules have been identified as being needed in GtkCAD.

The first type is the native GtkCAD library. These are libraries written specifically to extend the use of GtkCAD and make extensive use of its API and resources. They may be of little use to programs external to the GtkCAD framework.

The second type of plug-in is a library used to call an external program. An example would be a program written to be a stand-alone program such as a Finite Element Analysis program. The GtkCAD plug-in would then act like a translator and be used to write the appropriate input file for the program, and make a system call to start the program. The program would then run outside of the GtkCAD framework. During the run, the GtkCAD plug-in may provide monitoring functions. Upon completion, the GtkCAD plug-in may also provide some post processing analysis functions. This type of program execution is often referred to either as a wrapper or possibly middleware.

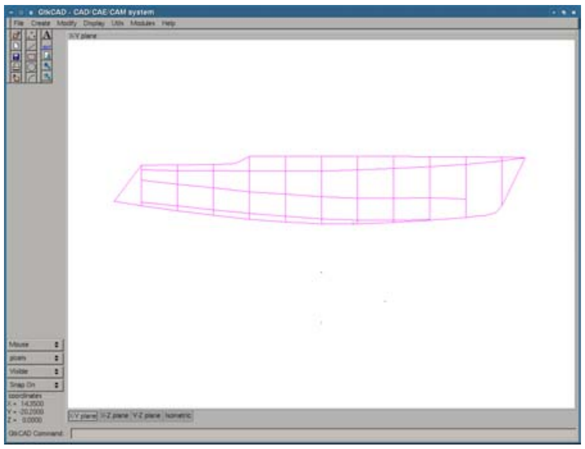

GtkCAD - CAD CAE CAM system
File Create Modify Display Utils Modules Help
X-Y plane
X-Y plane X-Z plane Y-Z plane Isometric
GtkCAD Command

Figure 5 GtkCAD Main Interface Window X-Y Plane View

In addition to these plug-in modules, as mentioned above, there are design mode modules used to modify the user interface to the needs of a particular engineering discipline. Examples of possible uses are the modification of new drawing dialogs, and import and export functions. These are needed because a new hull design in naval architecture mode may need a vastly different set-up than that of a new machine tool design in mechanical engineering, or that of architectural plans for a new building.

USER INTERFACE

The User Interface (UI) allows the user to interact with the program at runtime. This includes, but is not limited to, the main program window (when using a graphical window environment like X window, or Microsoft Windows), dialog boxes, menu structures, etc. It also includes the way the program receives data, for example from a mouse or keyboard.

Modern software is built with what are commonly referred to as toolkits for developing the UI. These toolkits are libraries of common widgets and a programming interface or API. This allows a quick and easy way to develop applications with a common look and feel, without having to reinvent the wheel for every application. GtkCAD uses the GTK+ toolkit developed by the Free Software Foundation for use with the GIMP drawing program. In fact, the name GTK is an acronym for the GIMP Tool Kit.

The interface of GtkCAD is based on the CAD programs that most designers are familiar with. While no particular interface is copied, most users would be productive using GtkCAD after just a couple of minutes of familiarization. In the near future, an on-line, context-sensitive help system will also be available.

The interface uses a notebook style widget to display the drawing areas as illustrated in Figures 4, 5, 7, 8. The first three tabs of the notebook displays a separate cutting plane, for instance the X-Y plane. The fourth tab is used for an isometric display, as shown in figure 4. Additional tabs may be added for additional displays. There is also a dialog window used to display a perspective view of the model, as shown in figures 6 and 9.

To avoid duplication of code, the UI tries to provide common functions to all modules, beyond the interface. For example, when a plug-in displays a graph, the graphing functions are provided by the UI.

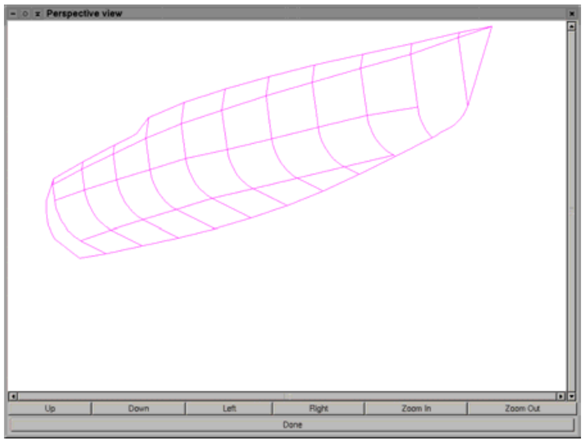


Figure 6 GtkCAD Perspective View Dialog Window

DATABASE

GtkCAD is designed to run in the GNU/Linux environment. This is a free implementation of the UNIX-like environment, which is a true multi-tasking, multi-user computing system. This means that it has the ability for multiple users to run the same program, at the same time, possibly from remote locations.

With this in mind, GtkCAD uses a SQL database for it data storage. The SQL database allows multiple users to query data from the database via transactions and data locks. The PostgreSQL database was chosen because it is licensed under the Free Software Foundation's General Public License (See http://www.gnu.org), implements the SQL-92 standard, has transaction processing, and implements both the embedded ANSI-SQL standard and a native library to include SQL statements within a C program.

One of the advantages of using a database for design data is that a vast majority of companies store their company data on SQL databases. This may mean that there is already in-house expertise on staff. By using a SQL database for GtkCAD's data storage it may be possible to link the project data with other client/company data, such as contact and/or billing information. While it is not within the scope of this thesis, it would be a simple extension of the Viewer application to include plug-ins for retrieving client contact information or billing details. Another possibility would be a simple stopwatch plug-in to automatically calculate billable hours on a project-by-project basis. By storing details of time spent in each design phase, future project timeline projections can become extremely accurate, avoiding future cost overruns.

The table structure within the database attempts to normalize the data structures used in the program. Program data is currently stored in a series of linked lists with pointers to data used by multiple lists and/or items, for example vertex points, colors, normal vectors, etc. Data with one-to-many relationships tend to be in their own table. For example a table for colors, vectors, and control points.

Drawing primitives are the next set of tables. Drawing primitives are basic drawing components such as points, lines, circles, etc. All primitives have common attributes. These attributes include color, size (pen width), line type, stipple, etc. They also may be grouped together to form parts, groups, etc. Alternatively, they may be drawn only in certain layers. These common attributes are placed in the entities table.

Each type of drawing primitive has its own unique characteristics. Therefore, each primitive has its own table to store the attributes unique of that drawing primitive. This table is related to the entities table using a one-to-one relationship.

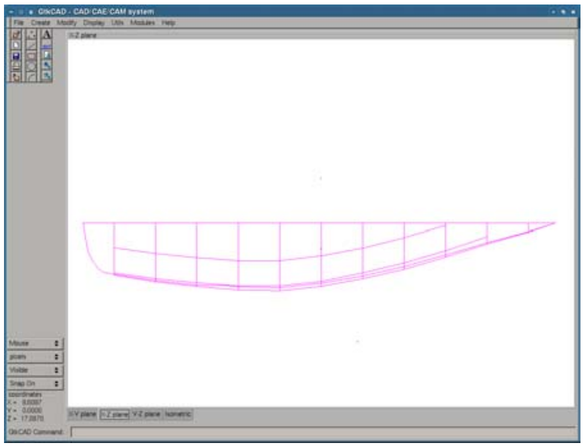
The final group of data tables are the support tables. Some drawing primitives have a one-to-many or many-to-many relationship with data. The support tables are used to specify the relationship between the two tables.

GRAPHICS

GtkCAD represents models using a library of graphics primitives to represent the edges defining a solid model. This describes an unambiguous representation of the model's surface, which then can be used for further analysis.

GtkCAD offers an API for the rendering of both two-dimensional and three dimensional graphics primitives, although only the 3-D graphics source code is currently under active development. The 2-D graphics environment utilizes the drawing utilities in the GDK library, a companion library to GTK+. The 3-D module uses the capabilities of the OpenGL graphics library developed by Silicon Graphics, Inc. (SGI).

GtkCAD actually uses the Mesa 3-D graphics library's implementation of the OpenGL library. This is a library licensed under the Free Software Foundation's General Public License (See http://www.gnu.org) that implements the OpenGL API, but is not licensed by SGI [www.mesa3d.org/about/intro.html].



coordinates
X = 8.0087
Y = 0.0000
Z = 17.2870

Figure 7 GtkCAD Main Interface Window X-Z Plane View

The data structures in GtkCAD were designed with four dimensions (x, y, z, w) in mind. This is also reflected in the vector definitions (i, j, k, l). For the purpose of this thesis the 'w' and the 'l' components are kept constant and not used.

The 2D graphics routines will be used to create the construction drawings needed by the builder. These drawings will be partially drawn automatically from the 3D model, and the designer will add any necessary details to the drawing of the model, such as notes and dimensioning placement.

SOLID MODELING

While this paper is not about Solid Modeling theory, the concepts involved are an integral part of any attempt at creating a three dimensional modeler for ship design. Solid modeling is part of the larger field of Geometric

Modeling and the related field of Computational Geometry [Patrikalakis].

Geometric Modeling is a field of mathematical representation of curves, surfaces, and solids [Patrikalakis].

Computational Modeling is the associated field of designing and implementation of the algorithms needed in geometric modeling [Patrikalakis].

A solid model is the digital representation of the geometry of an physical object, either real or envisioned. [Rossignac] From solid models, you can produce realistic renderings of the object, complete with lighting, shadows, and surface texture. They can also be analyzed for both geometric properties such as surface area, volume, moment of inertia, etc.. They may be assigned physical and mechanical properties, allowing structural analysis, fittings and interference. Typical examples of solid modeling are its uses in manufacturing and in development of Computer-Aided Manufacturing (CAM) programs, and robotic programming [Dewey].

Solid models have several characteristics that make them useful to engineers. These may include [Dewey]:

Three-dimensional: The model may be viewed from any angle.

Complete: The model is described sufficiently for analysis in all desired applications.

Unambiguous: The model ensures that the model is described sufficiently so that it is physically possible to create.

Systematic: The model is created and maintained using some sort of rational operation.

There are two major schemes used in solid modeling. The first is Boundary Representation, or BRep. BRep models can be thought of a wireframe model with a skin stretched over the resulting surface. The second is Constructive Geometry. Constructive Geometry is where solids are joined in boolean operations to form the final model. Constructive Geometry shapes are usually represented by machined shapes and thus commonly used in manufacturing environments.

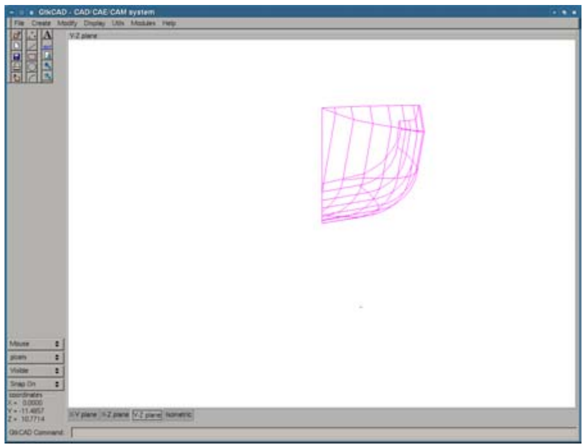


Figure 8 GtkCAD Main Interface Window Y-Z Plane View

GtkCAD and Solid Modeling Schemes

GtkCAD uses a Boundary Representation model, however only the geometry portion has currently been implemented. It is expected that future versions of the software will add the topology information. This should be a simple extension due to the fact that in a boundary representation, the geometry and topology information are often separate from each other.

Many of GtkCAD's plug-in modules such as Computation Fluid Dynamics and Finite Element Analysis require a Boundary Representation Model. Because the geometry and topology are separate in a BRep model it is relatively efficient to create the topology at later stages of the design, prior to analysis. Any design changes usually occur in the geometry and not the topology [Dewey].

GtkCAD uses the Computational Fluid Dynamics program, Gerris, which in turn uses The GNU Triangulation Library to describe solids in the computational domain. This library uses a simple format listing vertices, edges, and faces. The requirement is that the surfaces must be orientable, closed, manifold, and non self-intersecting, according to correspondence with the author Stephane Popinet.

Future work in GtkCAD will be to develop methods to ensure that these properties are maintained in the developed models.

Because the goal of creating GtkCAD from the beginning was to design ship hulls, it was recognized very early in the design process that Constructive Solid Geometry would not be appropriate. It is very hard to describe a ship's hull as a combination of common machining operations, but very easy when using such tools at Non-Uniform Rational B-Splines (NURBS), in a BRep representation.

PARAMETRIC MODELING

Parametric modeling is when a CAD system allows the user to specify relationships between two parts. The classic example is the door located in a wall. If the wall moves, then the door should move to remain within the wall with the same relative orientation and location.

A nice way to think about parametric modeling is as follows:

'With standard CAD systems when you change a part the dimensions change, while in Parametric modeling if you change the dimension the part changes.'

To understand this better, parametric modeling is based on locations not being absolute but rather related to a another point by some mathematical relationship. For example, in our door-wall case cited above, the door may be located within the wall by being 30% of the total length of the wall from the left corner. Now when the dimension of the wall changes, the door remains in the same relative position.

In naval architecture, an example might be the placement of bulkheads every 15 percent of the length between perpendiculars. Now if the vessel is lengthened or shortened that bulkheads remain in the same relative position.

GtkCAD uses the concepts of parametric modeling to describe the relationships between parts.

The current solution to the problem of parametric modeling is to use local and global coordinate systems. The model has a global coordinate system. The model is made up of an arbitrary number of parts, each defined around a local coordinate system. The local coordinate system is translated and rotated, about the global coordinate system, to obtain the correct position and orientation.

To demonstrate the concept, a simple program was written that displays a robot arm. The global coordinate is located at the shoulder. Local coordinate systems are located at each joint (elbow, wrist, fingers, etc.). The program allows the user to manipulate the arm, via the keyboard. As expected all movements are local to the proper coordinate system, so rotation at the elbow does not affect the shoulder or wrist [Woo].

The concept was also demonstrated by building a model of a sailing trimaran. The model consists of a main hull, two cross beams, an aka (outrigger), and the mast with standing rigging. The model is symmetric about the centerline, so only half the model was built.

An interesting future project would be to use a structural analysis program and explore how changes in position of the magnet points affect the structure. For example, if exploring rigging design in a sailing yacht, how would the rig be effected by moving the shroud attachment point 6 inches aft ? The program could analyze mast bend and the effects on sail shape much earlier in the design spiral and optimize design options.

DATA FORMAT

GtkCAD uses a Boundary Representation to define solid models. To create an unambiguous representation of the model, GtkCAD defines vertices, edges, and surfaces. Currently, GtkCAD's data primitives are mostly concerned with defining the geometry of a solid model, similar to a wireframe model.

Geometry

The most basic element of Boundary Representation models is the vertices. In GtkCAD vertices are defined as locations in three-dimensional space. They are used to describe the geometry of a model (physical size and location). Vertices are then linked into edges, which are used to describe the topology of the model.

It is also important to know directions in many engineering problems. This is provided by the use of vectors. Vectors in GtkCAD are defined in the normal mathematical sense, as a magnitude and direction, using the right-hand rule.

The geometry of the models in GtkCAD is made up of drawing primitives (points, lines, circles, NURBS curves and surfaces, etc.) called entities. Each entity has one or more attributes. These include color, size, stipple, etc. In addition entities are made up of other shared data structures such as vertices and vectors.

The entity is the most basic data block for all the data primitives. It contains all the common information needed by data primitives. This includes its id, name, color, stipple, and size.

Inheritance

Using the Object Oriented property of inheritance, all the other data primitives can be derived from this common parent. For example, a point contains most of the above data, plus the location of the defining vertex

Because it is unknown how many entities describe a model in GtkCAD, double linked lists are used to store the entities. This can be a rather inefficient method for searching and retrieval, so it will most likely be updated to a more efficient data storage method in the near future.

The data format in the SQL database and the machine's memory varies slightly. The SQL data tables have been normalized to eliminate the duplication of data in the various tables. This actually improves the performance of the SQL database. In the machines memory, the opposite is true. Searching for data in many tables is slow. GtkCAD data structures contain all the information needed, to eliminate searching.

Another difference in the database storage is the abundant use of id numbers to identify the various data structures. For example the id number 2 is used to identify GREEN. So a green point has a color id of 2 within the table row used to list the point.

GtkCAD currently uses a Cartesian coordinate system to define the geometry. The use of alternative coordinate systems, when appropriate, is being explored.

Topology

The topology describes how vertices are related to edges, edges to surfaces, and surfaces to volumes in a solid model.

Currently two of GtkCAD modules need topology information to run. These modules are the Computational Fluid Dynamics program, Gerris, and the Finite Element Analysis program, FELT.

GtkCAD requires a translator to convert the data format used by GtkCAD into a file format compatible to the plug-in program. In GtkCAD, this translator function is provided by a simple grid generator program. The transformation from the geometry data format native in GtkCAD to include the topology information is computed using the GNU Triangulation Library, GTS. This library can create the proper file from a map of the vertices defining the surface.

Initial work has been successfully completed that allows a GTS file of the exterior hull surface to be created from a hull form defined by a Cartesian grid. The middle body of the hull is unambiguously defined. Work needs to be done in the areas of the bow, and the stern toward the centerline.

LAYERS, PARTS AND GROUPS

In a typical design of anything more complicated than a brick, there is a tendency for drawings to contain too much information. In the days of paper drawings, this meant that multiple drawings were needed to divide the information into meaningful chunks of data. With today's modern CAD systems, this is accomplished with layers.

Layers are pieces of a drawing that can be turned on and off to control the amount and type of information displayed at any given time. For example, on a floor plan one layer might be the plumbing, the next the HVAC, and another the electrical layout. The plumber can turn off the HVAC and electrical layers, unless there is a need to check a fit in a tight corner. Likewise the HVAC and electrician can turn off the other unwanted information and concentrate on their own respective pieces.

Groups are used, as the name indicates, to allow the user to group common entities together. These groupings may allow for easy selection of entities during cut and paste operations, for example. They may also be used to simplify the creation of new parts (or blocks as they are called in AutoCAD).

Parts are used to break models up into components. These components then may be assembled together to create the end product. An example might be that the hull, mast, appendages, etc., are all individual parts. These parts are then assembled together to create a finished boat. Of course each part above is made up of many sub-parts, which may in turn be created from many sub-parts, and so forth.

The purpose of this design is to allow the individual parts to be drawn separately, and manufacturing data created. Yet allows the designer to check the part within the complete assembly for correct fit and tolerances.

The part data structure includes information on location and orientation, which is not needed in the group and layer structures. This allows the parts to be drawn in the correct relation to each other in assembly drawings. This is needed because all parts are drawn at the global origin, then translated and rotated to its correct local origin. This may need to be modified in the future as the use of parametric modeling is explored.

In the original design, the Layer, Part and Group information was incorporated into the entity data structure. This worked well for simple cases, but its limitations became apparent as models became more complex. For example, the original design limited the entity to be in a single layer, group or part. It also limited expansion into assemblies and other user defined organizations. To overcome these limitations, a new design was implemented that is extremely flexible.

The new design creates a separate list for layers, parts and groups. With this design the number of Layers, Parts and Groups are only limited by the number of available unsigned long integers on the given computer architecture. This also allows for new lists to be created for such things as assemblies and sub-assemblies.

There are also a series of corresponding lists that relate the entity data to the layer, part and group lists. These are called index tables. The use of index tables allow what is called a many-to-many relationships to exist between entities and layers, groups, and parts.

PROBLEMS AND FUTURE DIRECTIONS

To date the GtkCAD data format has been based on a generic geometric data type. This works well for drawing a very diverse set of physical objects. However there is a problem with this simple model when trying to evaluate a design via Computed Aided Engineering (CAE) techniques.

This is best illustrated by a simple example. Let's say ship's hull is being designed. The hull itself is easily described by the existing GtkCAD primitive data types, but to evaluate the ship's hydrostatics and stability more information is needed.

Some of this information may be calculated at run time. For example, the length on the waterline may be calculated by using bounding box techniques (i.e. looking at maximum and minimum values while keeping the draft constant). However, performing this calculation each time the hydrostatics are evaluated may lead to overly long response times from the software.

The necessary information could be input directly into the model data structures, but different data is probably needed by the various engineering disciplines. This leads to overly complex data structures, while to trying to satisfy the requirements for an unlimited mix of design needs.

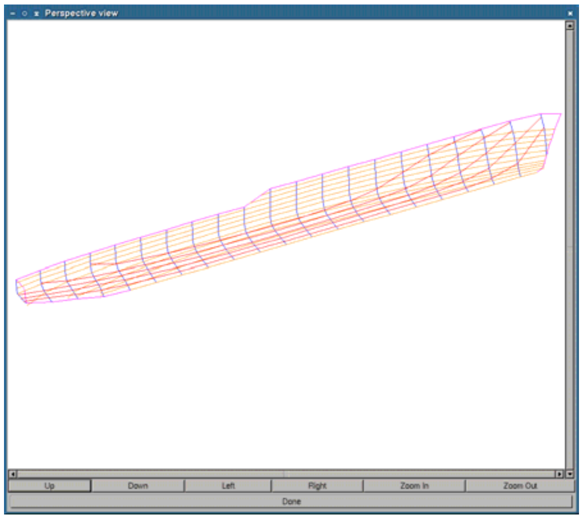


Figure 9 GtkCAD Perspective View of Destroyer Hull

The solution seems to be the use of object oriented programming technique of inheritance. This technique creates a common base data structure. As the model complexity increases, the new model inherits the traits of its parent data structure.

FUTURE DEVELOPMENT

GtkCAD is a program still under active development. Some of the methods used to implement the functionality in GtkCAD works but is inefficient. An example would be the use of linked lists to store the geometry primitives. While this is flexible and expandable, there are more efficient ways to store and retrieve the data. Linked lists are not very efficient in searches. A worst case search would require looking at every item in the list.

All the libraries that GtkCAD depends on are also under active development. Since the development of GtkCAD began, the GTK+ toolkit has had a new major release. This release is slightly incompatible with the old release. GtkCAD will need to be updated to use this new version of the GTK+ toolkit library.

More work needs to be done with the use on NURBS curves and surfaces. GtkCAD currently uses a polyline net to describe a surface. While this is simple and effective, it does not capture the true shape of the surface. However, the advantage of the current method is that the CFD plug-in module Gerris uses a polygon net, the GNU Triangulated Surface, to describe surfaces, so translation between the systems is straight-forward. It also is an advantage when creating a mesh for Finite Element Analysis.

One of the features that should be explored in the future is the use of thin clients. With thin clients, all the computational work is done on the server. This allows the use of disk-less work stations and other cheap terminals to be used by the end-user. This system is not without precedence, as many engineering workstations in the 1980's used this computing model.

New uses for GtkCAD should also be explored. For example, animation techniques used for accident recreation and investigation, game programming, etc.

SOURCE CODE AVALABILITY

The source code for GtkCAD is available on the GNU Savannah website:

http://savannah.gnu.org/

As with all Open Source projects, the success of GtkCAD depends on the participation of the user community. Please feel free to sign up to any of the available mailing lists and suggest improvements, and comment on where you feel GtkCAD needs to go to meet your needs. Also please use the bug tracking database to report any bugs, no matter how minor you feel they may be.

CONCLUSIONS

This paper has documented the development of an Integrated Design Environment for the Naval Architect on the Linux Operating System. It details the considerations that were important throughout the design process, and how the author dealt with the necessary design trade-offs.

Naval architects use the design spiral when designing a vessel. All the analysis that must take place for a successful design, uses many programs each of which may use its own incompatible data format, and different user interface that must be learned. By using a common user interface, and a common data format, GtkCAD has the potential to save many man-hours during the design process, while offering the flexibility to customize the analysis tools available, with a potential savings in the software budget.

GtkCAD uses the Open Source development model to provide a stable code base that all engineering disciplines can take advantage of, and allows the end-user the ability to customize the software as needed. The community also encourages the adaptation of proprietary software to be used as plug-in modules.

REFERENCES

[1] Edward V. Lewis; "Principles of Naval Architecture, Vols I, II, III" SNAME, Jersey City, NJ; 3rd Edition; 1988.

[2] Lars Larsson, Rolf E. Eliasson; "Principals of Yacht Design"; International Marine, Camden, Maine; 2nd Edition; 2001

[3] Stephen M. Hollister; "The Design Spiral for Computer-aided Boat Design"; http://www.newwavesys.com/spiral.htm; 1994

[4] Mason Woo, Jackie Neider, Tom Davis, and Dave Shreiner; "OpenGL Programming Guide, The official Guide to learning OpenGL version 1.2"; Addison-Wesley, Reading, Massachusetts, 3rd Ed.; 1999

[5] Prof. Nicholas Patrikalakis and Prof. Takashi Maekawa; "13.472J/2.158/1.128J/16.940J Computational Geometry - Lecture Notes"; http://ocw.mit.edu/index.html; Spring 2003

[6] Jarek R. Rossignac and Aristides A.G. Requicha; "Encyclopedia of Electrical and Electronics Engineering, Chapter Solid Modeling"; John Wiley and Sons; 1999

[7] Bruce R. Dewey; "Computer Graphics for Engineers"; Harper and Row, New York, NY; 1988

[8] Chinyere Onwubiko; "Foundation of Computer Aided Design"; West, St. Paul, MN; 1989

[9] Stephane Popinet and et alte.; "The GTS library Reference Manual"; http://gts.sourceforge.net/reference/book1.html; 2004

[10] Stephane Popinet and et alte.; "The Gerris Flow Solver Reference Manual"; http://gfs.sourceforge.net/reference/book1.html; 2004

[11] Stephane Popinet and et alte.; "The Gerris Tutorial, version 0.2.0"; http://gfs.sourceforge.net/tutorial/tutorial.html; 2004