Trident development roadmap

Prepared By:
Kathleen Svendsen

Martec Limited
1888 Brunswick Street, Suite 400
Halifax, Nova Scotia B3J 3J8

PWGSC Contract Number: W7707-145679/001/HAL

Contract Scientific Authority:
Neil Pegg, Head / Warship Performance, 902-426-3100 x165

Contract Report
DRDC-RDDC-2014-C311
June 2014

Lloyd's Register Marine

Working together for a safer world

Trident Development Roadmap

Martec Technical Report # TR-14-59

June 2014

Prepared for:

DRDC Atlantic, 9 Grove Street,
Dartmouth, Nova Scotia
B2Y 3Z7

Martec Limited
1888 Brunswick Street, Suite 400
Halifax, Nova Scotia B3J 3J8 Canada

tel. 902.425.5101
fax. 902.421.1923
email. martec-info@lr.org
www.martec.com

REVISION CONTROL

REVISION	REVISION DATE
Draft Release	March 31/2014
Final Release	June 3/2014

PROPRIETARY NOTICE

This report was prepared under Contract #W7707-145679/001/HAL, DRDC-Atlantic contains information proprietary to Martec Limited.

The information contained herein may be used and/or further developed by DRDC-Atlantic for their purposes only.

Complete use and disclosure limitations are contained in Contract #W7707-145679/001/HAL, DRDC-Atlantic.

SIGNATURE PAGE

TRIDENT DEVELOPMENT ROADMAP
Technical Report # TR-14-59 Rev 00
03 June 2014

Prepared by: Kathleen Svendsen **Date:** June 12/14
Kathleen Svendsen
Research Engineer, Field Services and Trident

Reviewed by: **Date:** June 5/14
Jim Covill
Team Leader, Field Services and Trident

Approved by: **Date:** June 5, 2014
David Whitehouse
Technical Manager

EXECUTIVE SUMMARY

The Trident Finite Element Software is a program suite that features capabilities for analysis and modelling of marine structures. This program has over a 30-year history but is in need of major revamping in order to best address the needs of the Royal Canadian Navy's modernization programs.

In order to optimally modernize Trident for this undertaking, it was determined that input from the current Trident user base was needed as a starting point. The survey findings were then used to conceptualize the next generation Trident product and finally, different technologies were examined to determine how best to achieve the desired product.

The survey was created and conducted with a representative portion of the Trident user-base. Surveys were predominately conducted in-person or on the phone to better facilitate a dialog.

It was determined that the major issues with Trident are user interaction; specifically

- the graphics engine performance and functionality,
- the layout and flow of the program menus and logic
- improvements to the documentation are necessary

Based on the survey findings, a technology investigation was conducted and a list of recommendations generated. Four different graphical user interface frameworks were investigated: Microsoft WinForms, Windows Presentation Foundation, Microsoft Foundation Classes, and QT. Of these frameworks, WinForms and QT offer the best results for effort expended. Four graphics engine libraries were also investigated; OpenGL, Visualization ToolKit (VTK), Direct3D, and HOOPS. The VTK and OpenGL graphical libraries are deemed the best options; but some additional investigation is recommended before a final decision is made.

Based on some quick prototyping, simply porting Trident's present graphical interface design to a modern technology does not solve some of the underlying UI issues. A more substantial change must be undertaken to create a better end product. Transforming the current Trident functionality into a dynamic linked library independent of all graphical routines should be the first step. This library should then be coupled to a new graphical user interface and graphics engine. The most promising and fastest route is to use Martec's Trident Visualizer which already has many common features with Trident and offers both a more modern GUI implementation and graphics engine. The Visualizer currently employs a WinForms GUI, and an OpenGL graphics engine.

It should be noted that while the Visualizer's graphics engine is already a major improvement over the current Trident system, some consideration should be given to a further upgrade. The open-source program ParaView offers possible resources and examples for implementation.

The current translators supporting NASTRAN, LS-DYNA, and ANSYS require enhancement. Users have also requested new features, some of which are: advanced geometric drawing/meshing capabilities, an export to Excel, an interface with Chinook and a with Solid Works translator.

The next phase of this undertaking is to begin the development of the next Generation Trident.

TABLE OF CONTENTS

LIST OF FIGURES

LIST OF TABLES

1.0 INTRODUCTION

The Trident program, for pre- and post-processing of finite element (FE) and naval/warship tailored analysis, has been developed by Martec Limited over the last few decades, beginning in the mid-1980s. Trident predates the Microsoft Windows environments and 3D graphics engines. The program has been significantly enhanced over that time period in terms of modeling and analysis capabilities; however, it is now in need of a concerted effort to upgrade and modernize the code, especially with regards to its graphics engine and graphical user interface (GUI).

This report details the results of the initial effort made to develop a formal roadmap for the next Generation Trident. The first phase of this undertaking was to design and conduct a survey of the Trident user base. The second phase was to explore various technologies and method implementations in order to select the optimal tools and techniques. The last phase was to generate a road-map for the next Generation of the Trident Finite Element Software.

2.0 TRIDENT SURVEY 2014

A user survey was developed to obtain feedback from a representative group of Trident users (Appendix A). The survey was conducted over the course of a month and was predominately conducted in-person or by telephone by Martec (Svendsen). Seventeen people in total were surveyed with an additional five people opting out due to limited current usage of Trident (see Table 2-1).

Table 2-1: List of Trident Users Surveyed.

Name	Department	Survey Type	Date Surveyed
Levi Morrison	Martec	In-Person	30/01/2014
Dustin Pearson	Martec	In-Person	30/01/2014
Phil Ruston	Martec	In-Person	30/01/2014
Brian Yuen	Martec	In-Person	30/01/2014
John Wallace	Martec	In-Person	31/01/2014
Iyala Koko	Martec	In-Person	31/01/2014
Brock Bolton	Martec	Phone	31/01/2014
Michael Abbott	Martec	Phone	31/01/2014
Ewa Pothier	Martec	In-Person	03/02/2014
John Mackay	DRDC	Phone	04/02/2014
Malcom Smith	DRDC	Phone	04/02/2014
Ian Thompson	DRDC	Phone	05/02/2014
Liam Gannon	DRDC	Phone	05/02/2014
Deanne Osmond	DND	Phone	06/02/2014
Johan Tuitman	TNO	Email	07/02/2014
Jan Czaban	NDHQ	Email	19/02/2014
Stefan Czaban	Babcock Canada	Phone	24/02/2014
Theo Bosman	Formerly of the Royal Netherlands Navy	Email	18/03/2014

Over the course of the surveys, several consistent trends emerged ranging from previously requested/suspected updates to Trident as well as other features that had not been previously considered in-depth. These findings will be biased towards issues users have found instead of benefits in order to maximize efforts in resolving the problems.

These findings are presented in prioritized order.

2.1 GRAPHICS ENGINE

The foremost enhancement requested by nearly all users is a better graphics engine. Currently a 3D model in Trident is difficult to manipulate and interpret, is slow in loading, and may crash. A fresh 3D display engine with more modern model manipulations such as rotation, panning, and zooming that can be done inherently with mouse control has been repeatedly requested.

2.1.1 Additional Graphical Features

Other display enhancement requests are element thickness, high quality rendering for picture exportation, viewing of nodes, and geometric modelling/meshing.

2.2 USER INTERFACE DESIGN

Trident is reported to have a steep learning curve and confusing design by many of those surveyed. An improvement to the user interface would be a better layout design and more intuitive flow has been requested by many users. There have also been appeals for custom tool bars, and hot keys.

2.3 DOCUMENTATION AND HELP

The majority of current Trident documentation is now outdated and documentation detailing new features is non-existent. Users have clearly expressed the opinion that this needs to be remedied. In addition, a theory manual should be written detailing the analysis the software is performing, the equations used, and significance of the resultant data representation.

There are limited tutorials for specific processes which are appreciated by those who were familiar with them. These users have requested more tutorials be written with specific focus on the more esoteric capabilities.

The current online help is generally disregarded, as much of it is found to be uninformative in providing further specifics about a feature. A more modern approach to help information has been requested such as context sensitive/cross-referenced or similar. Users have also requested that the documentation be more accessible since many have either missing documentation or broken links.

The majority of users rely on interaction with Martec staff for help or questions and this has been highly rated as having a ‘small entrepreneurial feel’ and is usually very quick to attain results.

2.3.1 Documentation of Code

Developers of Trident have also requested a more concentrated effort to comment the code, replace unlabeled numbers with named constants, and a more intuitive naming scheme

allowing for faster development. Some of this is inherent because of legacy code/old style FORTRAN. Code documentation should also include major functions, summaries of actions, and input/output data flow.

A better sorting of functions and the naming of their containing files would also improve ease of development, (ex. functions that manipulate strings are put in a file called string manipulation).

2.4 IMPORTING AND EXPORTING OF MODELS

Trident offers several different import/export translators and while these are greatly appreciated, users have stated many are outdated, error prone, or desire alternative translators.

2.4.1 NASTRAN

The NASTRAN translator is the most widely used at Martec and appears to be the best supported translator. However; there are still issues with it and users have reported it needs to be updated.

2.4.2 LS-DYNA and ANSYS

LS-DYNA and ANSYS are two other leading FE analysis software packages with Trident translator support, but reports of issues with the translations indicate both need to be updated. These two translators are highly used by a number of groups outside of Martec.

2.4.3 Chinook

Several users have requested a translator or interface with the Chinook CFD code be developed.

2.4.4 SolidWorks

A request has been made to implement a translator capability with SolidWorks.

2.4.5 Others

Other requests are the ability to export XY and contour plot information to Excel and geometric model information to VTK format (not to be confused with ParaView).

2.5 FILE STRUCTURE

A request has been made to reduce the file structure of Trident/VAST. This would mean using fewer files at runtime.

2.6 SPEED

Trident can be very slow when loading/importing and exporting large models (i.e. scalability issues). Several requests to reduce the time required of these processes have been made.

2.7 MEMORY USAGE

Ways to limit the amount of memory Trident can use have also been asked for. Often on smaller machines it allocates too much memory and slows down the machine drastically.

2.8 SUPPORT

Support for technical changes is extremely good; however, there are issues with non-technical change requests. There are also some communication challenges when the lead developer Merv Norwood is unavailable. Bugs in the software have been acknowledged but not fixed, request for better user interface; help information and feedback are not always available. When not for a specific contract there is often a lack of 'push' to get issues fixed.

2.9 OTHER REQUESTS

Other requests have been made for various new features and capabilities that will expand the Trident product both in quality and user friendliness.

2.9.1 Testing

It has been suggested that while VAST has a Regression Test-Suite, Trident does not. *A Regression Test-Suite for Trident would greatly improve the validation of current and new Trident capabilities as well as advance future development.* The test-suite would be a program that puts Trident through several tests and compares them to standard baseline.

2.9.2 Trident Wrapper

A wrapper to allow for command line usage, integration with other software (e.g. ShipRight) and ability to use with a scripting language (Python) has been requested by users/developers. This would allow for the usage of Trident to be more versatile and potentially gain support and a wider user-base.

2.9.3 Versioning

Versioning is a critical component and has been requested by several users/developers. Without versioning it is difficult to discern differences in Trident versions which may cause inconsistencies. A system of how and when new versions and licenses are released to users should also be implemented.

2.9.4 Animations and Video Export

The current Trident allows for basic animations but a request has been made for high resolution animations with the ability to record and export to video format has been made.

2.10 HIGHLIGHTS

While Trident has recognized usability issues, there are many capabilities and features that users have consistently mentioned are Trident's great strength. This has made Trident a unique piece if software that has no equal.

Its specialization in ships and marine structures make it more ideal to work with over more general purpose/expansive commercial software. Trident features a number of capabilities and data analysis that either is extremely expensive or non-existent in other commercial software. Its integration with the FE solvers VAST and NASTRAN give greater flexibility.

Several users mentioned that the Trident system analysis was often more reliable than other FE solvers and usually considered the more accurate analysis program in certain applications.

Trident also features a far smaller user base and so specialized analysis types and support from developers is far easier to attain.

Finally, there is one extremely significant advantage to the Trident product Suite – the entire source code is jointly owned by Martec Limited and the Canadian Crown with the majority of its development staff still employed by either Martec or the Canadian Department of National Defence.

3.0 TECHONOLOGY INVESTIGATION

The general trend of the survey indicates that the main focus for modernizing Trident should come in the form of improving the user's interaction with the software. This can be done in two steps, the first involving the manipulations of the 3D render of the model and the second with a more user friendly graphical interface.

To do this a number of different technologies and implementation styles were investigated in order to understand how to maximize effort spent in implementing these changes.

3.1 CURRENT CODE BASE

The current code base for Trident is FORTRAN. While newer FORTRAN 90 language constructs have been used over the past decade much of the programming style remains rooted in the older FORTRAN 77 procedural paradigms. A port to MS Windows 95 was achieved by using the MS FORTRAN 4 compiler and its supplied QuickWin interface. Amazingly, some

two decades later this interface is still both supported and enhanced with Intel FORTRAN 13.1. Unfortunately the graphics have become outdated in the fast paced computing world.

3.2 SIMILAR SOFTWARE

During the technology investigation a number of different programs were investigated. These programs offered insight into various capabilities, implementation methods, and technology usage. The two main programs investigated were the Trident Visualizer and ParaView.

3.2.1 Trident Visualizer

The Trident Visualizer was a program developed by Martec (Tom MacAdam) several years ago for post-processing of VAST analyses. The in-house program features communication with the Trident Database API and features its own graphics engine. The program also offers an array of post-processing capabilities and interaction with the 3D model. While this program's rendering performance needs to be improved for ever increasing larger FE model size, it offers a lot of potential as the starting point on which to base the new Trident.

3.2.2 ParaView

ParaView is open source software that is used for visually investigating large scientific datasets. Currently Trident allows for exportation of data to ParaView. This software has extensive capabilities in model viewing and interpretation from other program data files (i.e. NASTRAN, LS-DYNA, and ANSYS). These capabilities and the fact that all the code for the program is freely available makes it very interesting from either using its plug-in capabilities to put the functionality of Trident into ParaView or to use ParaView as a guide for Trident development.

3.3 GRAPHICS ENGINE

The graphics engine generates the 3D model rendering and performs user interaction with the model. Currently Trident generates its own model rendering by determining which pixels to color based on the models orientation and depth on the Z axis screen's component. While this was highly advanced during the initial creation of Trident it has since become outdated with the introduction of rendering pipelines and modern 3D modellers. There are several different graphical rendering libraries that were explored for using in the new Trident. The different graphic libraries investigated were OpenGL, Direct3D, Visualization ToolKit (VTK), and HOOPS.

3.3.1 OpenGL

OpenGL is one of the oldest and most established 3D rendering libraries. OpenGL is widely used in CAD, virtual reality, scientific visualization, information visualization, flight simulation, and video games. OpenGL is managed by the non-profit technology consortium

Khronos Group. It is multi-platform and allows developers to have complete control over the rendering. However; this does require more programming to preform modelling since it possesses a lower level functionality.

3.3.2 Direct 3D

Direct3D is part of Microsoft's DirectX application programming interface (API). This suite is highly versatile but is proprietary and Windows only. The DirectX software suite is predominately used for video games; however can be used for other graphical based engines. The DirectX runs mostly in native C++.

3.3.3 Visualization ToolKit (VTK)

Visualization ToolKit is an open-source library that is built on-top of OpenGL. It is multi-platform and is integrated with a number of different GUIs such as WinForms and QT. VTK has been extensively used in open-source programs such as ParaView and Image-Guided Surgery Toolkit. The library offers a wide range of high-level functionality while also allowing for users to modify their code and access the OpenGL routines. VTK also has built in capabilities for processing in multi-threaded rendering and dealing with large data sources.

3.3.4 HOOPS

HOOPS is a proprietary graphics framework produced by TechSoft3D. The HOOPS framework is used by several major finite element analysis programs and CAD drawing software. HOOPS can utilize both OpenGL and DirectX. Most of the support for this package is in the form of textbooks and purchasable documentation from SoftTech3D. HOOPS, unlike many graphic engines that are optimized for high-frame rates, is focused towards providing a visually precise representation.

Information on the usage of HOOPS is difficult to find other than their offered product information. There can be significant costs associated with HOOPS including licensing fees and developers support. Finally, a top tier priority support is available for developers; which includes training and technical expertise advice.

3.3.5 Conclusion

The short-term recommendation for a Trident graphics engine is to use the OpenGL based graphics engine in Trident Visualizer. This graphics engine has been already developed and requires less time to update and test.

It is recommended that for a long term solution a new graphics engine be built. Further investigation into graphic engine libraries should be conducted as well as possible collaboration with other programs requiring a new 3D modeller. It is suggested that at this time the Visualization ToolKit appears to be highly capable and features a good database of users, open-source example programs (ParaView), a high level of integrability, access to the lower level OpenGL, and a large set of development tools.

Table 3-1: Graphics Engine Technology Investigation Results.

	OpenGL	Direct 3D	VTK	HOOPS
Cost	Open-Source	Developer's Fee	Open-Source	Per-Copy
Base of Library	Multi-Platform	Windows	OpenGL	OpenGL/DirectX
Number of Users*	High	High	Medium	Unknown
Support*	High	High	Medium	Unknown
Ease of Implementation*	Hard	Medium	Medium	Unknown
Longevity*	Great	Great	Good	Fair
Licensing	Open	Microsoft	BSD license	TechSoft 3D
Language	C/C++	C++	C++	C, C++, C#, Java

* Anecdotal

3.4 GRAPHICAL USER INTERFACE

The current Trident implementation can be broken down into two main components. The parent window GUI serves as the main entry point to the program. There are also approximately sixteen hundred dynamic popup GUIs that gather input data. A number of GUI replacement technologies and implementation options were investigated.

3.4.1 WinForms

WinForms is an API included in Microsoft .NET Framework. WinForms is programmable in both managed C++ and C#. The WinForms is an event driven platform although it does not use the common Model-View-Controller framework. This library allows integration with most other libraries and quality graphical interfaces. It is compatible with OpenGL and VTK using the ActiViz.NET C# wrapper.

3.4.2 Windows Presentation Foundation

Windows Presentation Foundation (WPF) is part of the Microsoft .NET Framework. This library utilizes both C# and XAML to create high resolution graphics. It depends on the Direct3D graphics engine part of the DirectX library. This use allows for it to generate high resolution graphics. It is highly compatible with DirectX. WPF is one Microsoft's newest graphical user interface framework for desktop applications.

3.4.3 Microsoft Foundation Classes

Microsoft Foundation Classes is a Windows-based API that uses native C++. It is an older framework debuting in 1992. It was the basis of most Windows based applications and is still supported by Windows. This framework has been heavily used by Martec in the past and features quality graphics and high level of functionality. The framework does possess a higher learning curve. While Windows currently supports MFC there is no clear indication that it will

extend into the future and most MS software has migrated to other frameworks such as WPF and WinForms.

3.4.4 QT

QT is an open-source library software for developing graphical user interfaces developed by Digia. QT runs on most desktop platforms and some of the mobile platforms. It has extensive internationalization support. QT is available under a commercial license, GPL v3 and LGPL v2. QT was investigated through prototyping because its cross-platform capabilities, integration with VTK graphics library and Native C++ environment. It has high resolution graphics and is the graphical user interface for ParaView.

3.4.5 Conclusion

In conclusion, the most favorable GUI frameworks were WinForms and QT. WinForms was chosen since it is widely used at Martec, is implemented in some similar software (Trident Visualizer), and is not difficult to develop. QT was chosen since it offers a high compatibility with VTK, is open-source, offers a high level control over implementation, and possesses many examples of its implementation (ParaView). WPF was not chosen since there is not much familiarity with its style at Martec, there is a steep learning curve associated with its implementation, and inconclusive evidence that it offers high benefits compared to other frameworks such as WinForms. MFC was not chosen due to it having a large learning curve, lack of commitment by Microsoft, and offers little improvement over other frameworks.

Table 3-2: Graphical User Interface Technology Investigation Results.

	WinForms	WPF	MFC	QT
Platform	Windows	Windows	Windows	Multi-Platform
System	.Net	.Net	Win32	Open-Source
Development Effort*	Easy	Hard	Hard	Medium
Background of Developers	Extensively Used	Limited	Extensively Used	None
Licensing	Microsoft	Microsoft	Microsoft	GNU Lesser General Public License (LGPL) Or Corporate Licensing Agreement
Language	C#/C++	C#	C++	C++
Graphic Resolution*	Medium	High	Low	Medium
Longevity*	Good	Great	Fair	Fair
VTK Compatibility*	Good	Good	Fair	Great

* Anecdotal

3.5 IMPLEMENTATION OF TECHNOLOGY

To test if the selected technology offered the most coherent solution, quick prototypes were generated and tested with Trident.

3.5.1 Implementation of Graphics Engine

Two implementations of a new graphics engine were tested. A graphics engine that was previously built in OpenGL was tested for its capabilities with Trident and a quick prototype of VTK was also developed.

3.5.1.1 Implementation of OpenGL Graphics Engine

Currently the Trident Visualizer runs using an OpenGL based graphics engine (Figure. 3-1). This application is an improvement over the current Trident modeller. This graphics engine does require an update for newer model types and testing for large problems to ensure efficiency.

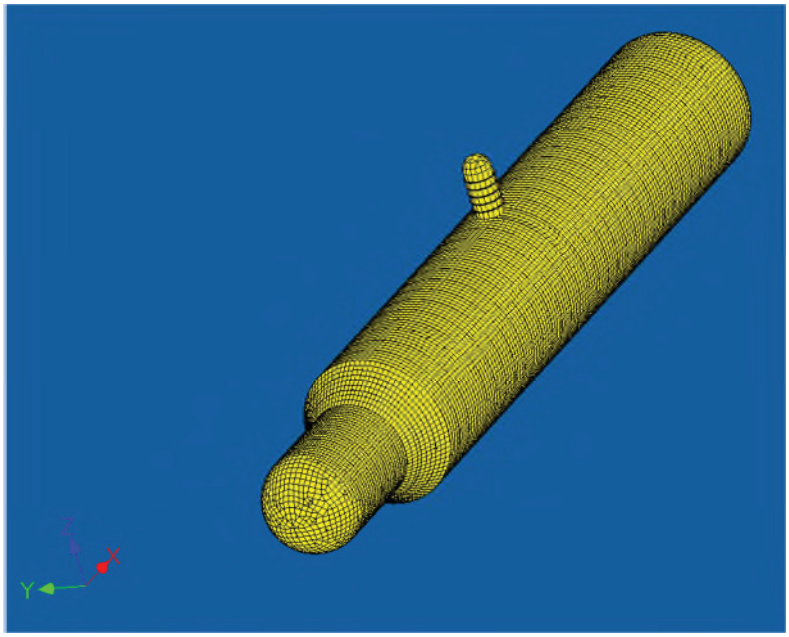


Figure 3-1 Trident Visualizer: OpenGL Graphics Engine

3.5.1.2 Implementation of VTK Graphics Engine

VTK was integrated into the current Trident as an off-shoot thread that reads data from the Trident Database API. This simple prototype showcased some of VTK's abilities. However, the VTK model was required to run in a separate window from Trident. There is a possible solution of 'gluing' the window into the current Trident but it would be inefficient and potentially hazardous. This would not be an issue if using a more modern GUI framework.

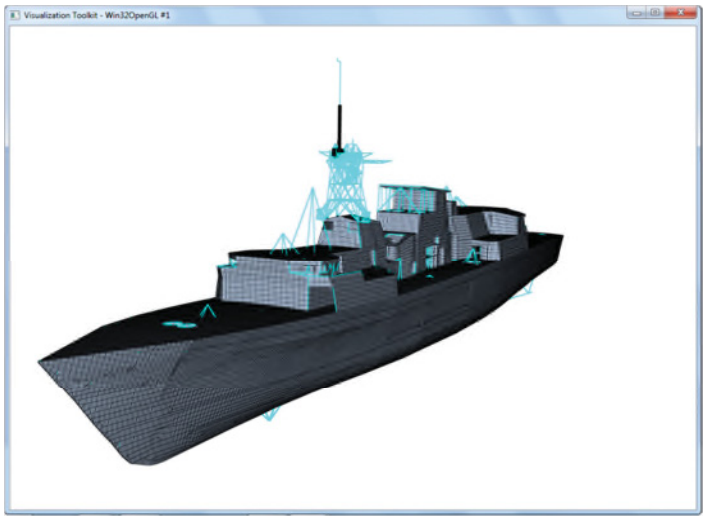


Figure 3-2 VTK Graphics Engine

3.5.2 Implementing newer GUI Technology into Current Trident

One of the first ideas to bring Trident up-to-date was to replace the pop-up GUIs with a more modern GUI implementation. This implementation had the old Trident spawn the newer GUIs. The first trial was done with WinForms C++.

3.5.2.1 WinForms

Two basic GUIs were developed using Winforms C++. These were implemented into the current version of Trident. The first was a new Working Directory (Figure. 3-3) popup that is used to specify the working directory at the beginning of the program.

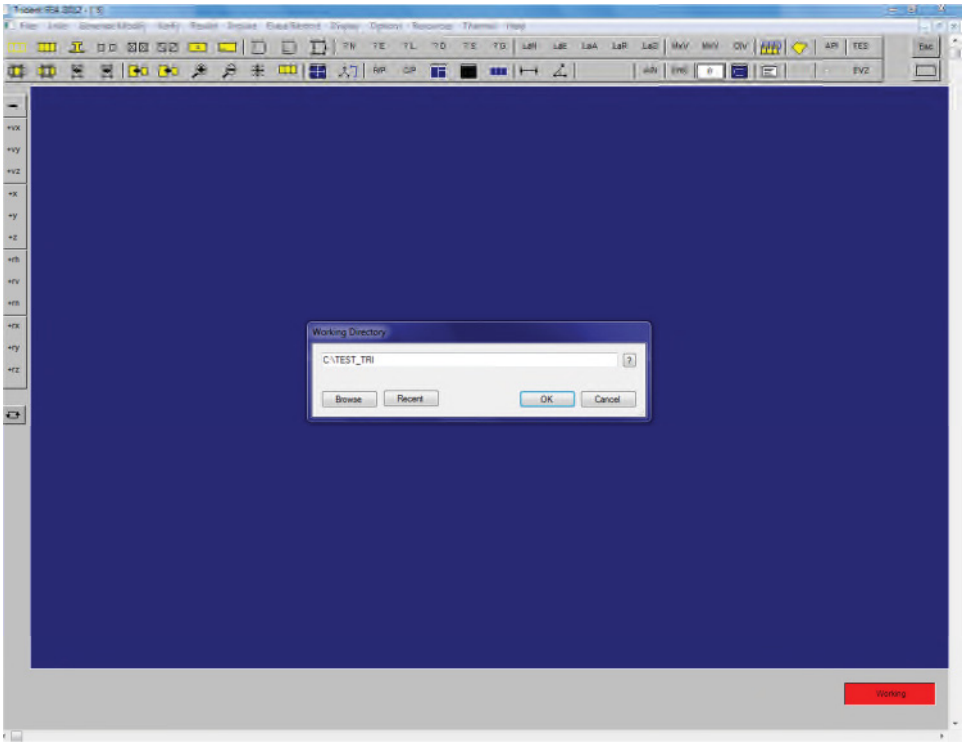


Figure 3-3 WinForms C++ Working Directory Popup

The second GUI that was generated was a dynamic GUI that took in a structure of data from Trident FORTRAN and used it to generate a specialized GUI (Figure. 3-4). This design allows for rapid development of GUI pop-ups to hasten the efforts to modernize the Trident software.

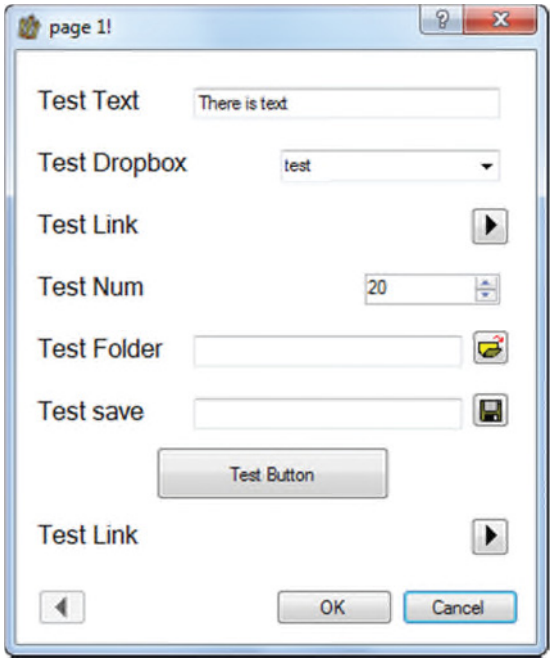


Figure 3-4 WinForms C++ Dynamic GUI Pop-Up

3.5.2.2 QT

The QT library was used to generate another Working Directory pop-up. The pop-up was integrated into Trident in the same method as the WinForms-based examples. The QT based window is shown in Figure. 3-5, the Trident window has been removed for clarity. Implementation of dynamic GUI elements (Figure. 3-4) in QT could be implemented with similar effort.

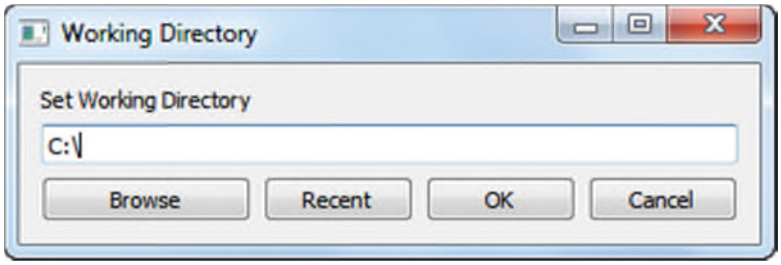


Figure 3-5 QT Working Directory Pop-Up

3.5.3 Implementing Current Trident into New GUI

While the newer GUI pop-ups were more pleasing to the eye they do not fix the underlying issue of confusing user interface and steep learning curve. Addressing this will require a different approach for managing the interactions between the user interface and the underlying Trident code. Currently the user-interface elements are invoked by the underlying Trident FORTRAN code. A more flexible/modular approach is to have the user interface call specific Trident functions as needed.

To investigate this approach the current Trident, hereafter referred to as the Trident library, was repackaged into a dynamic linked library. For a few specific functions (for testing purposes) the graphical routines were 'stripped' out of them. These functions were then exposed to being called by an outside source.

For fast prototyping the Trident Visualizer was used as a base with the new Trident library linked in. The Visualizer already deals with the Trident Database API which made it ideal for rapid prototyping with the new Trident Library. The Visualizer uses C# WinForms with an OpenGL graphics engine.

The Visualizer was programed to initialize the Trident library and run various functions such as open database, importing models and generate/modify nodes through the library similar to how old Trident would normally implement these actions. The Visualizer would then update its own information from the Database directly. This style allowed minimal changes to be made to the Trident library since it still 'thinks' it has a graphical interface while the Visualizer only tells the Library to run capabilities and waits until notified to update itself.

For tasks where pop-up dialogs are required by Trident, Callbacks were used. Callbacks are a method of coding where the setter (in this case the Visualizer) overwrites the addresses of function calls (the Trident library) redirecting the code to the alternative code. The library is unaware for the most part of this redirect and operates with little required changes.

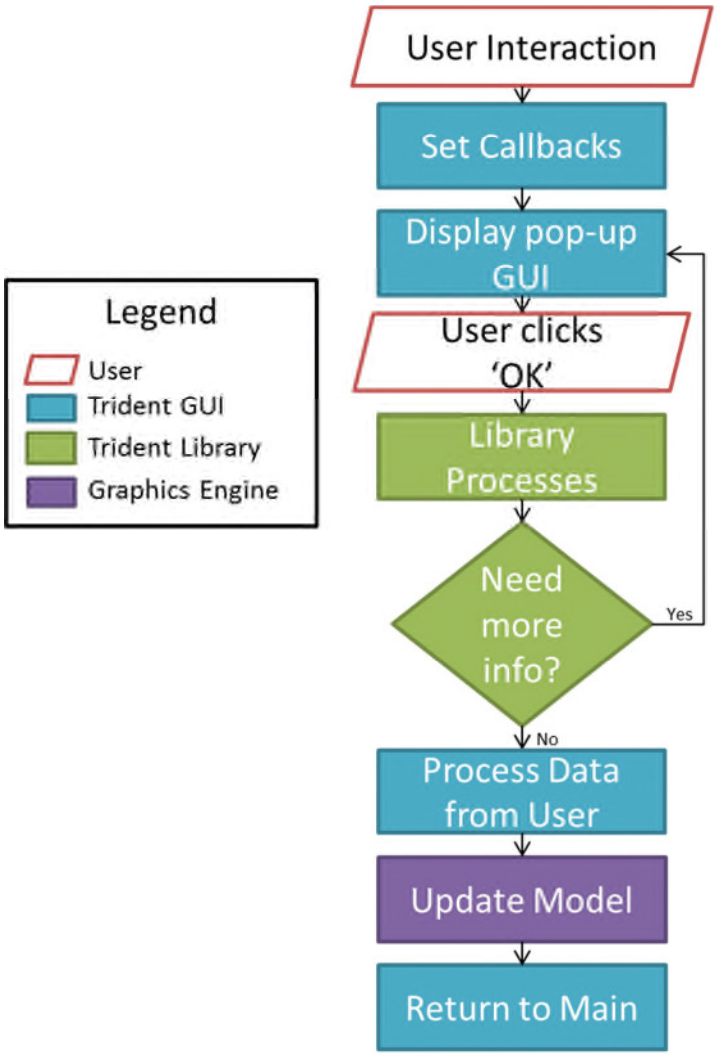


Figure 3-6 Implementation Model of New Trident

This transforming of the current Trident into a library and integration with the Visualizer was done successfully for a few of the generic but highly important cases: Initialization, Opening/Restart of Project, Closing of Trident Files, Import Model, and Creating/Modifying a Node (see Figure 3-7).

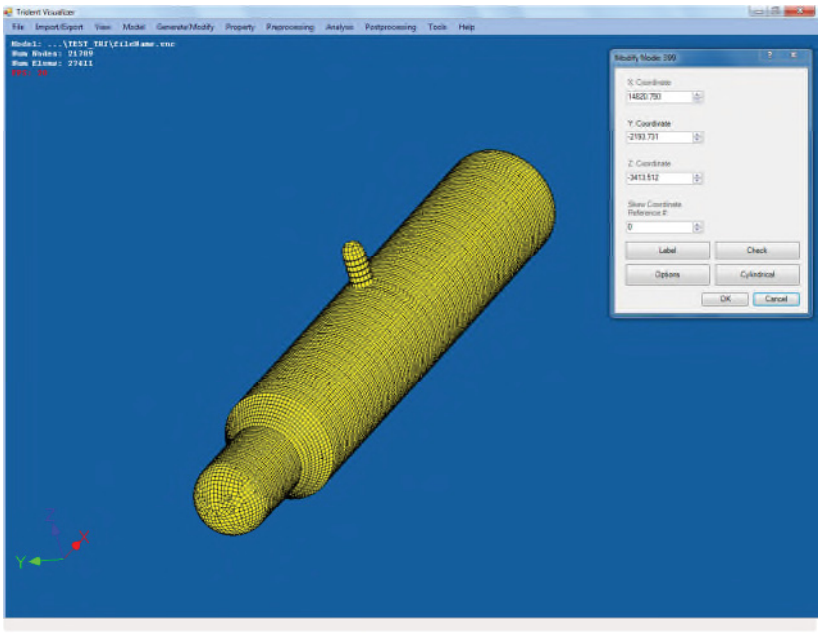


Figure 3-7 Trident Visualizer + Trident Library

This prototype indicated that this approach was highly viable as a solution to the modularization. Majority of the Trident functions act in three similar fashions: basic info from user, interaction with model, or a combination. The aforementioned test functions form the basis of almost all interactions used in Trident. This approach also minimizes the changes to the Trident code since it almost solely requires the graphic routines to be overwritten, not the main functions.

4.0 NEXT GENERATION TRIDENT – THE WAY FORWARD

As clearly indicated by the surveys, Trident is a highly capable program but significant efforts must be expanded to ensure it does not become obsolete. The following comments and recommendations are presented to pave the way for the modernization and advancement of Trident.

4.1 DEVELOPMENT MANAGEMENT

Trident is a very large program and the following recommendations are more generic in nature and pertain to more long-term tasks to support and improve development of Trident.

4.1.1 Source Control

There currently is no shared source control for Trident. This must be the first step before major work begins in the new Trident Development. This will prevent loss of work, and allow for more coherent tracking of issues and completed work.

4.1.2 Versioning

Currently there are numerous versions of Trident with little to no obvious differences between them. This causes confusion and prevents both other developers and users from having the most stable version. Versioning should become critical to the development with set release dates that are pushed to all the applicable clients. This will allow for users to know what version they have and be able to keep track of updating. This needs to be auditable with a record of versions and changes between them.

4.1.3 Regression Test-Suite

In order to maintain Trident an auto-tester should be built with a series of tests and expected results. This will allow for developers to be able to easily test all of Trident ensuring everything is working smoothly. Debugging statements should also be placed through the code in Trident using pre-processer define statements to maximize the information given to the developer during debugging/testing.

4.1.4 User List

A maintained list of all users, their license's expiry date and how to contact them should be made for easier administration and communications.

4.1.5 Line of Communication and Tracking of Issues

Currently there is no clear path for users to request changes or report bugs. While most go directly to head developer Merv Norwood, when he is not available there is no clear line of communication. There is also no tracking of issues and a lack of feedback to users regarding the raised issue.

As such it is recommended having a set of instructions for reporting issues with a backup if the main line is unable to be reached. All reports/requests should be set in a central tracking with a deadline to be looked into. All closed request must be reported back to the user that reported it.

Trident requires a bug tracking system, currently LR uses VI. If a critical bug is fixed all users should be informed and the new fixed Trident be pushed to clients.

4.1.6 Post Development Support

Trident's main focus over its development life to date has been on its capabilities with many client based contracts adding to its core functionality. It is strongly recommended that a post development support strategy be formalized for the purposes of continued debugging/upgrading, documentation, and response to user requests. This will allow developers to fix issues in Trident or add/modify features to better enable the user.

4.2 NEW CODE ARCHITECTURE

In order to advance Trident it is recommended to re-architect the program into a more common modular format. Currently Trident is a fully integrated GUI, Event Scheduler, Graphics Engine, and Capabilities. These Capabilities incorporates its translators, database API and other special functionalities. It is these Capabilities that make Trident so important and the goal of modernizing Trident is for these features to be maintained and not lost. These are generic aspects and need to be addressed so that Trident usage can expand beyond its current, limited user base.

The new implementation would use an event driven design with the main program built on a graphical user interface with the specific capabilities of Trident being an external library. The graphical engine would also be its own library allowing an easier progressing and eventual replacement in the future.

The next Generation Trident should be as modular as possible which will improve future development and prevent Trident becoming obsolete.

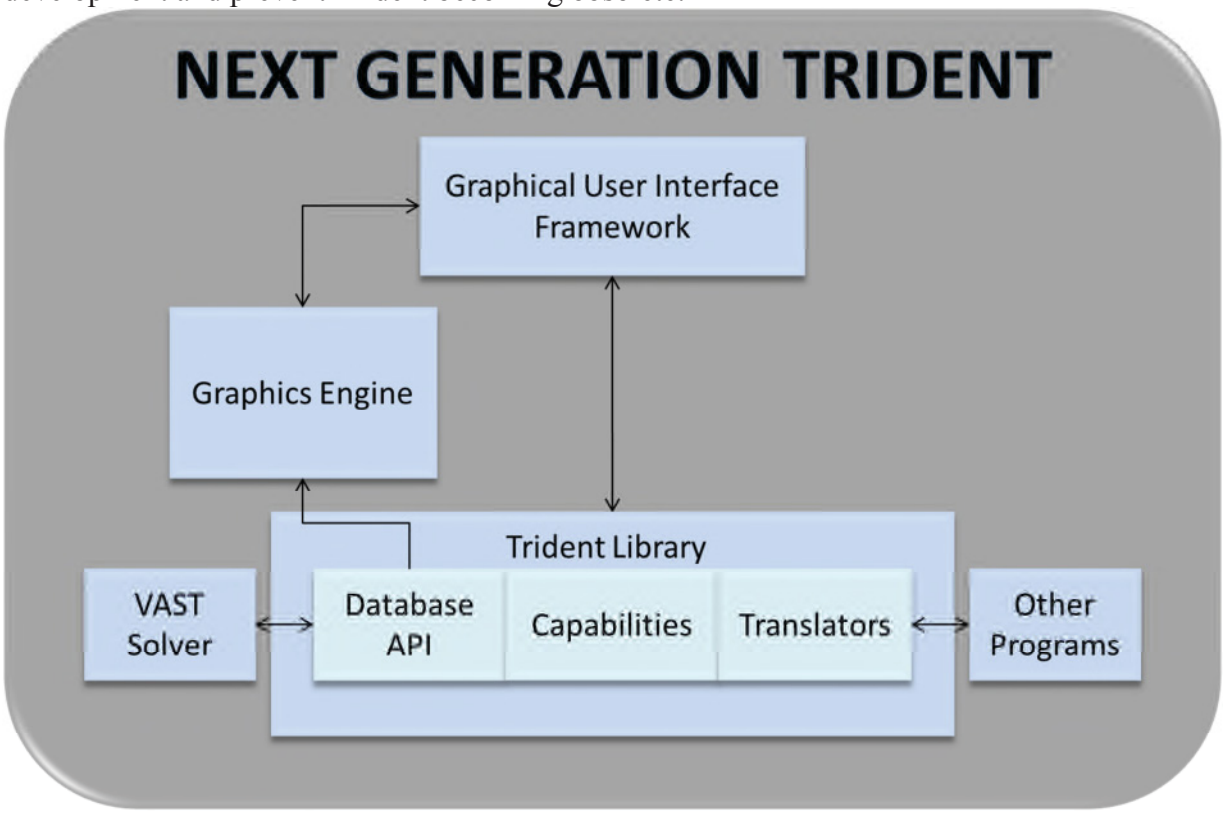


Figure 4-1 Trident Next Generation Framework

4.2.1 Trident Library

It is proposed to remove the user based functionality from 'Trident' and reform the Capabilities into a dynamic linked library that is independent of the user interface. A new user

based functionality based program will then use the Trident library to create the next Generation Trident.

The Trident Library would allow for the Trident Capabilities to be 'Black Boxed' making them highly versatile both for future integration, modernization and expansion. The library would act using a number of exposed function calls that could be called from front-end applications such as graphical user interfaces or even a command line application. The library could also act as a plug-in for other programs such as Chinook or ShipRight.

4.2.1.1 Code Cleaning

Trident is nearing thirty years old and has expanded over the years but possess code that has become redundant and unnecessary. The code is largely undocumented, with most variable naming schemes using the old FORTRAN six letter limit. This lack of documentation and proper naming schemes can make it difficult in interpreting code. Unlabeled numbers that hold no obvious meaning are also common throughout the code. All of this makes it difficult for new programmers to understand and modify the code.

With the approaching retirement of many of the original developers it is vital to both have documentation in place describing the code as well as new developers trained on the code so that the standard of capabilities can be maintained.

It is recommended that during the process of developing the Trident library the code should also be appropriately commented/documented, unneeded data/code removed, naming schemes improved, and unlabelled numbers be replaced with named constants. While this will not have immediate benefits the effects will allow for the continued expansion and development of Trident.

4.2.1.2 Library Wrappers

Creation of library wrappers in languages such as C and Python will allow for a wider usage of the Trident library creating a more versatile framework.

4.2.1.3 Specify Component Types

Users should have the ability to tag components with names. These tagged names could help with analysis, sorting through large assemblies and specific analysis on single parts.

SubSAS requires named components but models will lose the tags when ported to Trident causing issues and corrupted data.

4.2.2 Trident Graphical User Interface

It is recommended to use the Trident Visualizer as the basis of the new Trident Graphical User Interface. This will allow for rapid development since it already features an OpenGL graphics engine, communication with the Trident Database API, and many post-processing procedures.

This front end will communicate with the Trident library by sending it commands and user information and then update itself from the database based upon the actions that the user took.

4.2.2.1 User Interaction

It is recommended that during the revamping of the Trident Visualizer, user input should be incorporated into the layout and flow of the GUI. This will allow for better design implementation minimizing the learning curve.

4.2.2.2 Features

The additions of features designed for ease of use will both lower frustrations amongst users and increase productivity. Some features that have been suggested are hot keys, customizable toolbars, and improved toolbar graphics.

4.2.3 Graphics Engine

The current Trident graphics engine is 2D render that was developed several years ago in FORTRAN. The modeller is the leading complaint of all users surveyed and the awkwardness of rotating, zooming and other basic model manipulations is conveyed to be very frustrating.

4.2.3.1 Trident Visualizer OpenGL Engine

Currently the Trident Visualizer uses an OpenGL based graphics engine. This graphics engine is an improvement in terms of graphics quality, ease of use, and speed over the currently used Trident FORTRAN graphics engine.

However; to use the Visualizer's graphics engine for Trident it needs to be updated and thoroughly tested for larger complex models. This solution would allow for a rapid development of the new Trident since it would be quicker to update this graphics engine to use for the initial Trident release while a newer one is being developed.

4.2.3.2 Trident Next Generation Graphics Engine

It is recommended for a long term solution that a new graphics engine is developed within the next few years in order to stay current and competitive in the market. The new graphics engine could be developed for multiple programs with special features implemented with Trident. This graphics engine could also use open-source programs such as ParaView as a guide in development since it exhibits many useful features such as slicing planes, and blueprinting of slices.

During the investigation on new graphics engine libraries the two most promising appeared to be OpenGL and VTK. VTK being interesting since it offers a higher level approach to OpenGL with many useful additional capabilities. As such it is recommended that VTK be further investigated as a solution for the next generation graphics engine.

4.2.4 Translators

There are numerous different Finite Element solvers available and most users of Trident work with different ones as well. Being able to translate information from one solver to the next is critical and while Trident offers many different Translators; these may be out of date causing issues for users in the form of lost data or corrupted models.

4.2.4.1 NASTRAN, ANSYS, LS-DYNA

Almost all users of Trident use one or more of the NASTRAN, LS-DYNA, or ANSYS finite element analysis programs. Possessing a high level of compatibility with these programs will ensure faster and higher quality of results for users. It is recommended to put a concentrated effort into updating the current translators or research into an alternative solution such as third party software.

4.3 NEW FEATURES

Several new features were identified during the user survey. These are potential ways to enable Trident to continue to grow and meet clients' needs. These features are possible future projects but are considered to be out of scope of the current proposed project but are mentioned for consideration and documentation.

4.3.1 Templates

It is recommended that templates should be designed and implemented. These would consist of common components/structures. Users could be able specify various geometric and meshing details and then the newly created component could then be 'snapped' into place. Users could then modify the component to meet their requirements. This would allow for better efficiency of the user's time and decrease time for modelling.

4.3.2 Geometric Drawing/Meshing

It is recommended that alterations be made to allow users to create/modify a model using geometry based tools. These would allow for easier and faster model refinement.

When using a graphical based user interface a representation of the new part should be displayed to the user before they confirm adding it to the model.
An example of the possible geometric drawing/meshing is:

- Draw a line and extrude a plane from it.

4.3.3 Excel

It is recommended that an export to Excel format for plotting based data be made. The Excel format is cross compatible with many programs and allows users versatility in viewing their data.

4.3.4 Chinook

It is recommended that an interface with Chinook CFD be implemented. This has been requested by several different users and would add to the capabilities and resources of both programs.

4.3.5 Solid Works

It is recommended that an import of Solid Works models be developed. Solid Works is a widely used model generator. Currently users of Trident have to remake models almost completely when importing them from Solid Works causing frustration and lost time.

4.4 DOCUMENTATION

There exists very little current documentation on Trident. It is recommended that an appropriate effort be put in generating useful documentation for users and developers. This information could be made through an external application similar to Visual Studio Help. This program would contain theory, tutorials, key terms, menu items overview, and GUI features such as hot keys and toolbars.

4.4.1 Theory Documentation

Theory documentation will serve as a reference for users to validate their results from Trident. This information will contain all mathematical equations and high level overview of analysis. It should tie in with tutorials.

4.4.2 Tutorials

Currently Trident has a few tutorials that are highly rated. The next generation Trident should include a large array of tutorials focussing on common problems for new users and more rare analysis types since these are less likely to be fully known. These tutorials should be made part of the searchable help for best results.

4.4.3 'Did You Know' Quick Tips

A 'Did You Know' quick tip section could display tips at start-up and list them for users to look through. These tips would offer users insights into features they may not be aware of. They would be useful to make users more aware of various capabilities of Trident and a way to improve their knowledge of Trident.

4.4.4 Toolbar/ GUI Quick Tips and Help Access

Various help information on toolbars should be accessible through right click. GUIs should offer a feature to instantly access the help information that is associated with its function.

4.4.5 Menu Organization

The help should also contain information on all the buttons/menu items the user is able to access describing what they are used for and linking to tutorial/analysis type connections.

4.4.6 About

Information on the version of Trident, libraries, help application, licensing and contact information for issues/questions would also be included in the help application and Trident Main.

4.5 DEVELOPMENT TEAM SUGGESTIONS

In order to maximize the progression both for the new Trident and for future development more developers are needed. The following recommendations include ideas for recruitment for Trident Developers.

4.5.1 More Developers Trained on Trident

Currently Trident is predominately supported through a limited number of developers. This causes issues if they are unavailable or busy with other contracts. More developers trained on Trident would allow for faster response time in fixing bugs or developing new features.

4.5.2 Students

The development of the new Trident will be a massive undertaking and with the majority of current developers having other commitments the hiring of students would allow for faster development.

4.5.2.1 Undergraduate Co-op Students

While it is important not to rely on students or put them on critical path tasks the hiring of co-op students could allow for faster development and a fresh perspective of Trident. Undergraduate co-op students could act in many of the less theoretical tasks such as documentation of the code, testing, and adding higher level features such as hot keys and toolbars.

It is recommended to hire students with a more mathematical and programming background, English skills would be critical for documentation. A recommended group would be Electrical, Mechanical Engineering and upper year computer science students. Computer Science students would be a particular good fit when developing the new Graphics Engine since there is a larger focus in their degrees on similar applications.

4.5.2.2 Graduate Students

Graduate students specifically in the field of Finite Element Analysis would be a great resource for developing the new Trident. They would be highly useful in theory documentation and dealing with the more technical aspects of Trident. These students would also be useful in designing the tutorials since they would have the basic understanding of finite element analysis but would offer insight into usability and difficulty level.

5.0 DEVELOPMENT SCHEDULE

Developing the next generation Trident is an extensive and important undertaking. To optimize its implementation, it has been separated into three main tracks, three recommended tracks, and one optional track. This project spans two fiscal years and is divided into quarterlies. The first three tracks are instrumental to the new Trident development and include project management, planning, and implementation of the new Trident. The next three tracks are for the test framework methodology, user documentation, and theory documentation. The final track is for improvements made to the graphics engine.

A detailed budget spreadsheet with timeline projections is included as Appendix B, but is summarized below.

5.1 TASK 1: PROJECT MANAGEMENT

This task would span the length of the project and would mainly be for keeping the project on track and organizational purposes. This task is estimated to be 384 hours.

5.2 TASK 2: PLANNING

This task would take approximately one and a half quarters to complete and would take place in the first fiscal year. The planning would involve cataloguing the various functionalities of Trident, and prioritizing for their implementation with a bias towards functionality for support of CSC. This task is estimated to be 400 hours.

5.3 TASK 3: TRIDENT IMPLEMENTATION

This task constitutes the main body of the project and encompasses the actual implementation of the Next Generation Trident. This implementation is broken into various phases over the two years. There are six phases which will begin roughly in the second quarter of the first year and finish at the end of the project. This task is estimated to be 4560 hours.

5.3.1 Phase 1: Setup

This phase would be of putting in place any required setup, prioritization of the implementation and allow for support code to be implemented.

5.3.2 Phase 2-4: Implementation

These three phases would be the actual implementation of the various Trident capabilities into the new framework. Each phase would have a checklist determined in the planning track which would allow for a record of capabilities that need to be added and have been completed. The first phase would comprise of the essential capabilities while the third phase would comprise of useful but of a less immediate need. Each of these phases would ideally take a quarter and a half of implementation time so approximately four and a half months. Therefor the entire implementation phase would take about 12 months split over two fiscal years.

5.3.3 Phase 5: Beta Testing

This phase would see initial testing of the new Trident. The primary focus would be on fixing any critical bugs and streamlining the user experience. Modifications based on user feedback would take place to improve the next generation Trident.

5.3.4 Phase 6: Release

This phase would see the end of the project in which all the release files would be packaged for distribution to users.

5.4 TASK 4: TEST FRAMEWORK

This task is divided into two phases and is for developing a test framework to improve the quality of Trident, reduce development time, reduce bugs, and decrease testing time for the next generation Trident. The first phase would be designing the logistics for the testing framework and implementing its skeleton. The second phase would be populating the testing suite in correlation with the Trident capabilities. This task is estimated to be 280 hours.

5.5 THEORY DOCUMENTATION

This task is for completing the theory documentation of the computational analysis that Trident performs. This document will ideally be used as a reference and learning tool for understanding what and how Trident achieves its tasks. These documents will also be useful in teaching new users the principles behind Trident and its different analysis capabilities. This task is estimated to be 960 hours.

5.6 GRAPHICS ENGINE UPDATE

This task is for updating the 3D modeller to include some new features, improve model manipulation, decrease loading time to the renderer, and improve overall graphic quality. This task is completely optional but is recommended for a better user experience and quicker user time. This task is estimated to be 160 hours.

6.0 CONCLUSION

A representative group of Trident users were surveyed to gain a better understanding of Trident usage and future development needs. Results of the survey conveyed a need to significantly improve user interaction with a focus on the graphics engine, layout/flow logic, and online/reference documentation. Based on these results, a technology investigation was conducted. Martec's Trident Visualizer and KitWare's ParaView programs provide additional design insight into the next generation Trident. The most promising graphics technologies are either OpenGL or VTK (Visualization ToolKit) with the new user interface being implemented via Microsoft WinForms.

From an architectural perspective, to truly improve and modernize Trident requires a new framework and a higher modularization of its components. This necessitates a conversion of the old Trident routines into a callable library that would be devoid of any user interaction. This library would then be integrated into the Trident Visualizer which already features a (less dated) graphics engine and desirable user features. This modular program would then become the next generation Trident.

The three main tracks for the next generation Trident are:

1. High Level Project Management
2. Planning
3. Implementation

The three highly recommended tracks are optional but essential for the next generation Trident:

1. Test Framework
2. User Documentation
3. Theory Documentation

The final track is recommended but optional:

1. Graphics Engine Modernization

A rough project timeline was generated along with estimate completion time for the tasks. The tasks are mostly broken down into phases mapped to quarterlies that will be completed over two fiscal years.

APPENDIX A

TRIDENT SURVEY 2014

Trident Survey 2014

Date: / /2014 (dd/mm/yyyy)

Surveyor: Kathleen Svendsen

User:

Notes:

1. Number of years of experience you have using Trident.
    - Trident: ____ years

2. Why do you use Trident

3. What do you like most about Trident?

4. What are the main components of Trident you use?
    - Static and Dynamic Analysis
        - Linear
        - Non-Linear (-Geometric, material, strain-rate effects)
        - Time Integration
        - Modal Superposition
    - Natural Frequency
    - Buckling Analysis
    - Foray Frequency
    - Life 3D
        - Crack Propagation
        - Fatigue
        - XFEM
    - UNDEX
        - Whipping
        - USA/FSI (Fluid Structure Interaction)
            - Whipping
            - DAA
            - CASE
    - DDAM/ BR3021/ SRS
    - Post processing
        - XY Plots
        - Animations
        - Contour plots
        - Export Bitmaps
    - Static Balance
        - Trim
        - List
        - Draft
    - Mass Distribution
    - Ship Specific Functionality
        - Moment of Inertia of a cross-section
        - Fluid Elements
        - Center of Gravity (COG)
    - Model Generation/ Refinement
    - Import/Export of Third Party Software

- Other:

- Do you find they meet all your needs?

- What tools would you like to be added?

5. Does Trident meet your current and future needs?
    - Current

    - Future

    - If Not what needs must be fulfilled for you?

6. Do you find the control layout/flow intuitive?

7. What are your top three complaints regarding Trident?(More if needed)
    - Complaint 1:

    - Complaint 2

    - Complaint 3

8. Do you have issues in figuring out how to use Trident?

9. We are aware that the resource files are out of date but do you find the resource files, online help and manuals useful to you?

    - If not what would you find useful to help you use Trident.

10. What would cause you to stop using Trident?

11. What other FE or visualization tools do you use (or wish to use) with Trident?

Use: Want:

12. What other FE tools do you use instead of Trident, how long and why?

- Tool:
  - Reason:
- Tool:
  - Reason:
- Tool:
  - Reason:
- Tool:
  - Reason:
- Tool:
  - Reason:

13. Would additional capabilities to generate Finite Element Meshes in Trident be useful to you?

- Would this be a significant impact?

14. Is the fact that Canada/Martec owns the source code significant to you?

15. How do you feel about the quality and speed for service in regards to Trident?

16. And how do you rate Trident on a scale of 1-5 with 5 being good versus other tools:

• Functionality (what Trident can do USA,etc)	1	2	3	4	5	N/A
• Use of Third party software	1	2	3	4	5	N/A
• Range of solver capabilities (VAST, other)	1	2	3	4	5	N/A
• Selection of Translators (NASTRAN)	1	2	3	4	5	N/A
• Range of Post-Processors (xy plots, etc)	1	2	3	4	5	N/A
• Quality of Results (post processing)	1	2	3	4	5	N/A
• Number of Bugs (severity)	1	2	3	4	5	N/A
• Layout Design (i.e. to finding where functions are)	1	2	3	4	5	N/A
• Ease of Use (is it easy to use?)	1	2	3	4	5	N/A
• Graphics –(user interface)	1	2	3	4	5	N/A
• Graphics –Modeler(i.e. 3d render)	1	2	3	4	5	N/A
• Speed (not the solvers)	1	2	3	4	5	N/A
• Speed (the solvers)	1	2	3	4	5	N/A
• Learning Curve	1	2	3	4	5	N/A
• Capability (as a whole)	1	2	3	4	5	N/A

- Overall Design 1 2 3 4 5 N/A

17. As a whole how happy are you with Trident
    - Frustrated
    - Annoyed
    - Indifferent
    - Happy
    - Love it!
    - Other:

18. Do you know of anyone else who uses Trident currently?

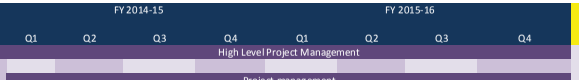
Name: Department: Phone/Email

19. Do you have any other issues or comments?

APPENDIX B

NEXT-GENERATION TRIDENT ROADMAP, PHASE 1- BREAKDOWN

Next-Generation Trident Roadmap, Phase I - Core Architecture and Functionality (Draft)



			Who	Labour	Travel	FY 1	FY 2	
High Level Project Management								
Throughout	Project management	Managing teams, admin, etc.	PM	384		192	192	Total (hr.)
Sub-Total (h)				**384**				
Sub-Total ($)				**$ 62,077.44**	**$ -**	$ 31,038.72	$ 31,038.72	Total
Track 1 - Planning			**Who**	**Labour**	**Travel**			
	Planning	Catalog required functionality to support CSC	MEN	80		400		Total (hr.)
		Meet with LR (SH) on common framework planning	MEN, TAM, DW (JDC)	96	$ 12,000.00			
		Gap analysis with existing FE codes	MEN, ENG	80				
		Detailed re-architecture plan (prioritized)	DEV	80				
		Project planning	JDC, TAM	32				
		Put versioning infrastructre in place (version numbers, source control)	KAS	32				
Sub-Total (h)				**400**				
Sub-Total ($)				**$ 64,664.00**	**$ 12,000.00**	$ 76,664.00		
Track 2 - Implementation			**Who**	**Labour**	**Travel**			
Phase 1	Setup	Put in place UI framework that will support re-architecture work (e.g. TV)	DEV	80		80		Hours
		Commence prioritized Trident re-architecture and integration in new UI framework	DEV	120		120		
Phase 2	Implementation 1	Continue prioritized Trident re-architecture and integration in new UI framework	DEV	960		960		
Phase 3	Implementation 2	Continue prioritized Trident re-architecture and integration in new UI framework	DEV	960			960	
Phase 4	Implementation 3	Complete prioritized Trident re-architecture and integration in new UI framework	DEV	960			960	
Phase 5	Beta	Internal and external acceptance testing	ENG, TST	960			960	
		Address issues	DEV	480			480	
Phase 6	Release	Package for release	KAS	40			40	
Sub-Total (h)				**4560**		1160	3400	Total (hr.)
Sub-Total ($)				**$ 737,169.60**	**$ -**	$ 187,525.60	$ 549,644.00	Total
Track 3 - Test Framework			**Who**	**Labour**	**Travel**			
Phase 1	Framework design	Implement skeleton test framework	DEV, TST	40		40		Hours
Phase 2	Populate tests	Populate test framework	TST	360		135	225	
Sub-Total (h)				**400**		175	225	Total (hr.)
Sub-Total ($)				**$ 64,664.00**	**$ -**	$ 28,290.50	$ 36,373.50	Total
Track 4 - User Documentation			**Who**	**Labour**	**Travel**			
Phase 1	Framework design	Implement skeleton help framework	DEV, HLP	40		40		Hours
Phase 2	Document	Populate help framework	HLP	240			240	
Sub-Total (h)				**280**				
Sub-Total ($)				**$ 45,264.80**	**$ -**	$ 6,466.40	$ 38,798.40	Total
Track 5 - Theory documentation			**Who**	**Labour**	**Travel**			
Phase 1	Write theory manual	In consultation with developers, write theory manual	DOC	960		369	591	Total (hr.)
Sub-Total (h)				**960**				
Sub-Total ($)				**$ 155,193.60**	**$ -**	$ 59,689.85	$ 95,503.75	Total
Track 6 - Graphics engine modernization			**Who**	**Labour**	**Travel**			
Phase 1	Planning	Assess need and survey suitable options; decide on best way forward	SW	40		40		
Phase 2	Implementation	Implement changes to UI framework without affecting Trident development (Track 1)	SW	120		80	40	
Sub-Total (h)				**160**		120	40	Total (hr.)
Sub-Total ($)				**$ 25,865.60**	**$ -**	$ 19,399.20	$ 6,466.40	Total
				Labour	**Travel**	**Total**		
Total (h)				7144			2456	4688 Total (hr.)
Total ($)				$ 1,154,899.04	$ 12,000.00	$ 1,166,899.04	$ 409,074.27	$ 757,824.77 Final Total