Tyche 2.3

Supporting Documentation

Pawel Michalowski
Contractor

DRDC CORA CR 2009–005
October 2009

Defence R&D Canada
Centre for Operational Research and Analysis

Maritime Operational Research Team
Centre for Operational Research and Analysis

National Defence Défense nationale

Canada

Tyche Version 2.3

Supporting Documentation

Pawel Michalowski
Contractor

Prepared By:
Pawel Michalowski (Brainhunter (Ottawa) Inc.)
Brainhunter (Ottawa) Inc.
1545 Carling Avenue Suite 600
Ottawa, ON Canada K1Z 8P9
Software Developer
PWGSC Contract Number: EN537-8-4015/182/ZL
CSA: Cheryl Eisler, Maritime OR Team, 613-990-3737

The scientific or technical validity of this Contract Report is entirely the responsibility of the Contractor and the contents do not necessarily have the approval or endorsement of Defence R&D Canada.

Defence R&D Canada – CORA

Contract Report

DRDC CORA CR 2009-005

October 2009

Principal Author

Original signed by Pawel Michalowski

Pawel Michalowski

Contractor

Approved by

Original signed by Dr. R.E. Mitchell

Dr. R.E. Mitchell

Maritime Section Head

Approved for release by

Original signed by Dale Reding

Dale Reding

Chief Scientist

Defence R&D Canada – Centre for Operational Research and Analysis (CORA)

© Her Majesty the Queen in Right of Canada, as represented by the Minister of National Defence, 2009

© Sa Majesté la Reine (en droit du Canada), telle que représentée par le ministre de la Défense nationale, 2009

Abstract

Tyche is an application that was developed by the Maritime Operational Research Team in 2004. It is a stochastic force structure model that simulates the demand placed on future maritime fleets. After creating a realistic set of missions, fleets and fleet assets are scheduled to respond to the given circumstances. However, Tyche simulations take time to set up and can be computationally quite expensive. To improve both speed and the user's ability to administer and maintain a queue of simulations, a new Tyche Simulation Engine (Tyche SE) module has been derived from the existing Tyche application. By sharing the same code base while restructuring the architecture, a background process version of Tyche (Tyche SE) is used solely for the purpose of running automated simulations through the Tyche simulation management tool dashboard. To maintain backwards compatibility and debugging functions, simulations can still be run from the Tyche graphical user interface (GUI). The GUI now includes several new features and code improvements, including compiled help files.

Résumé

Tyche est une application développée par l'Équipe de recherche opérationnelle (Mer) en 2004. C'est un modèle de structure de force stochastique qui simule la demande imposée aux futures escadres maritimes. Après la création d'un ensemble réaliste de missions, les escadres et leurs ressources sont programmées pour répondre dans des circonstances données. Toutefois, la préparation des simulations Tyche peut exiger beaucoup de ressources de calcul et de temps. Pour augmenter la rapidité et la capacité de l'utilisateur à administrer et à tenir à jour une file d'attente de simulations, un nouveau module Tyche SE (*Simulation Engine*) a été dérivé à partir de l'application Tyche existante. En utilisant la même base de code, mais en restructurant l'architecture, on a obtenu une version de traitement en arrière-plan de Tyche (Tyche SE) qui n'est utilisée que pour l'exécution de simulations automatisées au moyen du tableau de bord des outils de gestion des simulations de Tyche. Les simulations peuvent encore être exécutées à partir de l'interface utilisateur graphique (GUI) Tyche dans le but de conserver la compatibilité ascendante et les fonctions de débogage. L'interface comprend désormais plusieurs nouvelles fonctions et des améliorations du code, notamment des fichiers d'aide compilés.

This page intentionally left blank.

Executive summary

Tyche Version 2.3: Supporting Documentation

Pawel Michalowski; DRDC CORA CR 2009-005; Defence R&D Canada – CORA; October 2009.

Background: In 2005, an investigation of fleet size and the capabilities required by the future naval fleet was undertaken on behalf of the Canadian Navy by the Defence Research and Development Canada (DRDC) - Centre for Operational Research and Analysis (CORA) Maritime Operational Research Team (MORT). This work, referred to as the Fleet Mix Study (FMS), was completed using Tyche - a computer program that is maintained, developed and validated in-house. The software is still employed today as a highly useful tool for the Navy, but additional pre-and post-processing tool development was required. At the same time, restructuring the code to a model-view-controller (MVC) software architecture would prepare the software for further upgrades. Several deficiencies were also identified which needed to be rectified without delay.

Results: Tyche 2.3 improved the user interface by eliminating existing behaviour anomalies, as well as adding several new features. In more detail, enhancements to Tyche version 2.3 included:

a. Resolution of multiple program crash issues from the previous version (2.2) by means of system-wide Global Error Handling and Logging. Implementation of proper error handling throughout entire Tyche code base prevents it from crashing unexpectedly. Now, when an error occurs, it is displayed on screen, logged in a file, and the user can continue using the application.

b. Improved support for log files to record progress and diagnose simulation problems. As a result of these changes and additions, Tyche is now able to handle a much larger volume of simulations with greater reliability and flexibility. The advent of proper error handling allows the user to continue using Tyche without interruption (“crashing”), while at the same time logging problems with depth of detail down to the code module, which allows for faster identification of problems.

c. Integrated user support through help files. Compiled HTML Help (CHM) files facilitate better understanding of the Tyche applications by the end user.

d. A search function for the Operational Schedule (OpSched) Viewer. Implementation of a search capability in the OpSched Viewer to find Scenarios and correspondingly assigned Assets, or to find levels of Asset usage, eliminates the need for opening a large text file and searching manually. This feature also includes the saving of search results into a text file.

e. Separation of Simulation Engine (SE) from Tyche GUI. In order to further improve the efficiency in which Tyche simulations are performed, a new executable module was created independent of the user interface. This SE module runs as a background process in Windows, called from the command line interface, and is used in conjunction with Dashboard for running unattended simulations.

Significance: Tyche provides a means for efficient access to information generated by a simulation model that schedules assets within a fleet to address a set of mission scenarios, based on a set of rules that emulate a real fleet scheduler. The input/output and generation of data processes have been streamlined. Simulation data can be easily accessed, modified and reviewed. Numerous bugs have been fixed.

Future plans: A number of code enhancements can be made to speed up data entry and simulation data generation. One of the possible technical improvements could be converting the Visual Basic 6.0 code base into a faster processing code and then further adapting it for use in a parallel processing environment. Plans have been developed to increase the scope of problems that Tyche can handle by allowing more user-defined input/output and greater algorithm flexibility.

Sommaire

Tyche Version 2.3: Supporting Documentation

Pawel Michalowski; DRDC CORA CR 2009-005; Defence R&D Canada – CORA; October 2009.

Contexte: En 2005, l'Équipe de recherche opérationnelle (Mer) (ERO(M)) du Centre d'analyse et de recherche opérationnelle (CARO) de Recherche et développement pour la défense Canada (RDDC) a entrepris une étude portant sur la taille et les capacités de la force navale du futur pour le compte de la Marine canadienne. Ces travaux qui portent le nom d'Étude sur la composition de la flotte (ECF) ont été effectués au moyen de Tyche, un logiciel dont la maintenance, le développement et la validation sont effectués à l'interne. Ce logiciel est encore utilisé aujourd'hui et il constitue un outil très utile pour la Marine, mais il a été nécessaire de développer des outils de prétraitement et de post-traitement. Par ailleurs, la restructuration du code dans l'architecture modèle-vue-contrôleur (MVC) préparerait le logiciel en vue de lui apporter des mises à jour ultérieures. D'autre part, plusieurs lacunes qui devaient être corrigées sans retard ont aussi été identifiées.

Résultats: Tyche 2.3 a amélioré l'interface utilisateur en éliminant les anomalies comportementales existantes et en ajoutant plusieurs fonctions. Voici plus en détails les améliorations apportées par la version 2.3 de Tyche :

a. Résolution de plusieurs problèmes de plantage du logiciel de la version précédente (2.2) au moyen de processus globaux de traitement et de journalisation à l'échelle système. La mise en œuvre de processus de traitement des erreurs approprié dans l'ensemble du code de Tyche l'empêche de planter inopinément. Maintenant, lorsqu'une erreur se produit, elle est affichée à l'écran et inscrite dans un fichier, et l'utilisateur peut continuer d'utiliser l'application.

b. Une meilleure prise en charge des fichiers journaux permet d'enregistrer le déroulement des simulations et de diagnostiquer les problèmes connexes. Ces changements et ajouts ont pour résultat que Tyche est maintenant en mesure de traiter un volume de simulations beaucoup plus important, tout en présentant un degré de fiabilité et de souplesse plus élevé. L'ajout d'une prise en charge des erreurs convenable permet de plus à l'utilisateur de continuer à utiliser Tyche sans interruption (causée par le plantage du logiciel), alors que la journalisation des problèmes en détails jusqu'au niveau du module logiciel permet une identification plus rapide des problèmes.

c. Soutien intégré de l'utilisateur au moyen de fichiers d'aide. Les fichiers HTML compilés d'aide (CHM) favorisent une meilleure compréhension des applications Tyche par les utilisateurs finals.

d. Fonction de recherche pour le module de visualisation des calendriers opérationnels (OpSched). La mise en place d'une fonction de recherche dans le module de visualisation des calendriers opérationnels afin de trouver des scénarios et les actifs connexes qui leur sont affectés ou encore de trouver les degrés d'utilisation des actifs élimine le besoin d'ouvrir un gros fichier texte et d'effectuer une recherche manuelle. Cette fonction permet aussi d'enregistrer les résultats des recherches dans des fichiers texte.

e. Séparation du moteur de simulation (SE) de l'interface utilisateur de Tyche. Afin d'améliorer l'efficience de l'exécution des simulations Tyche, un nouveau module exécutable indépendant de l'interface utilisateur a été créé. Ce module SE s'exécute en arrière-plan dans Windows, il est appelé à partir de l'interface à ligne de commande et il est utilisé avec le Tableau de bord afin d'exécuter des simulations sans supervision.

Signification: Tyche constitue un moyen donnant un accès efficient à l'information générée par un modèle de simulation qui effectue l'affectation des actifs d'une flotte afin d'exécuter un ensemble de scénarios de mission en se fondant sur un ensemble de règles opérationnelles qui émulent un véritable planificateur de la flotte. Les processus d'entrée/sortie et de génération des données ont été simplifiés. On peut facilement accéder aux données, les modifier et les examiner. De nombreux bogues ont été résolus.

Plans futurs: Plusieurs modifications peuvent être apportées au code afin d'accélérer la saisie des données et la génération des données de simulation. Parmi les améliorations techniques envisageables, on pourrait convertir la base de code Visual Basic 6.0 en un code de traitement plus rapide et adapter le résultat afin de l'utiliser dans un environnement de traitement parallèle. Des plans ont été élaborés afin d'élargir la gamme de problèmes qui peuvent être pris en charge par Tyche en permettant un degré plus étendu de personnalisation des entrées/sorties et en assouplissant les algorithmes.

Table of contents

List of figures

Acknowledgements

The author would like to acknowledge the assistance of Cheryl Eisler and Alex Bourque from the Maritime Operational Research Team. They provided assistance with code interpretation, functional testing, and documentation development. Ms. Eisler, in particular, developed significant portions of the help text from reports she and previous Tyche developers authored or co-authored.

This page intentionally left blank.

1 Introduction

1.1 Background

In 2005, an investigation of the size and capabilities required for the future naval fleet was undertaken on behalf of the Canadian Navy by the Defence Research and Development Canada (DRDC) - Centre for Operational Research and Analysis (CORA) Maritime Operational Research Team (MORT). This work, referred to as the Fleet Mix Study (FMS), was completed using Tyche - a computer program that is maintained, developed and validated in-house.

Tyche is a stochastic simulation model that schedules assets within a fleet to a set of mission scenarios, based on a set of rules. Tyche version 1.0 was initially developed for scheduling and fleet planning, and has proven to be a valuable tool for naval configuration explorations to respond to the investigation requirements posed by the Fleet Mix Study [4]. However, certain assumptions made by Tyche (version) 1.0, such as fixed asset crews and capabilities, limited the Tyche simulation model. These limitations warranted the development of Tyche 2.0, which also possesses fundamental differences in some of the modelling and scheduling methods.

Tyche 2.1 further addressed limitations of the Tyche model, such as the inability to correctly model the current concept of operations for sealift and personnel tempo constraints. The general inability to handle large volumes of simulations and the absence of managing a simulation prompted the development of Tyche 2.2.

Tyche 2.2 addressed the following: an improved graphical user interface, creating a more modern appearance and increased user friendliness; the ability to run simulations unattended in the background while consuming only idle computing power; support for log files to record progress and diagnose simulation problems; a new simulation editor to streamline simulation setup; a centralized multi-simulation management tool, the Tyche Dashboard, to issue commands to simulations and maintain a queue; the ability to resume simulations that were terminated or otherwise interrupted by software or hardware failure(s); and resolution of multiple program crash issues from the previous version.

After Tyche 2.2 was used for the second iteration of the Fleet Mix Study [5], areas for development were identified to improve the user experience and help prepare the software for significant future upgrades.

1.2 Objective

In order to more efficiently manage the software development, the services of a computer programmer contractor were required. The roles of the contractor were to perform routine software maintenance/upgrades, develop integrated help file support for the software, add functionality to post-processing tools, and to modify the software architecture for future use in a parallel processing environment. Each of these items will be discussed in turn. The purpose of this report is to document the development of each function by the contractor.

1.3 Scope

This document is not intended as a stand-alone user guide for Tyche. It documents an incremental change in the design of the program, and illustrates to programmers and users the approach taken. Previous documents that are still in effect include references [1]-[3]. For a comprehensive user guide, please refer to the newly created help files for Tyche, which can be accessed through the software itself.

2 What’s new in Tyche 2.3

2.1 Outline

There are four distinct areas of improvement completed for Tyche Version 2.3:

1. **Global Error Handling and Logging** - implementation of proper error handling throughout the entire Tyche code base. Now, when there is an error it is displayed on screen, logged in a file, and the user can continue using the application. The global error handlers provide improved support for log files to record progress and diagnose simulation problems. As a result of these changes and additions, Tyche is now able to handle a much larger volume of simulations with greater reliability and flexibility. The advent of proper error handling also allows the user to continue using Tyche without interruption, while at the same time logging problems with depth of detail down to the code module, which allows for faster identification of problems.

2. **Integrated Help Support** - development of a compiled HTML Help (CHM) file to facilitate better understanding of the Tyche applications by the end user.

3. **Search function for the Operational Schedule (OpSched) Viewer** - implementation of a search capability in the OpSched Viewer to find Scenarios and correspondingly assigned Assets, or to find levels of Asset usage, eliminates the need for opening a large text file and searching manually. This feature also includes the saving of search results into a text file (isolating only the data of interest).

4. **Separation of Simulation Engine from Tyche GUI** - in order to further improve the efficiency in which Tyche simulations are performed, a new executable module was created independent of the user interface. This SE module runs as a background process in Windows, called from the command line interface, and is used in conjunction with Dashboard for running unattended simulations.

In terms of implementation, global error handling now prevents Tyche from crashing whenever an unhandled error is generated, which is then logged into the Windows Event Viewer. An XML file is also generated to record the errors raised in the GUI, the same functionality as implemented in the Dashboard for simulation runs. Help support is evidenced in the existence of a help file available within the Tyche program, as well as context-sensitive help links throughout. The search function for the OpSched Viewer is an extension of the GUI form to allow users to search for specific assets or scenarios that occurred during a simulation. The search results are automatically displayed in a tree view hierarchy based on iteration, or can be exported to a text file. The user will not notice any difference in functional performance by the splitting of the simulation engine from the main GUI; however, this will facilitate further performance improvements in a high-throughput computing environment (refer to Section 0 for more information). Development and operation of each function will be discussed in the following sections.

Further additions or modifications introduced into Tyche 2.3 that do not warrant in depth discussion here include:

- Modification to the Ideal Asset Selections form. Upon right-clicking a cell in this window, Tyche now displays the level number, base and list of matched capabilities for that particular asset during the particular scenario.[1] Two bugs associated with that functionality have also been fixed;
- Changing of the column background color from black to white is now working on the first click when the OpSched viewer is first opened;
- Elimination of the flashing scroll-bar in the OpSched Viewer (when the user scrolls up or down);
- Colour coding dialog boxes no longer crash when modifying colours of scenarios in the OpSched Viewer;
- Tyche no longer disappears when trying to open a new file after closing a currently open file without saving;
- The Capability Supply lists for Asset Types now allow the user to enter new Capability requirements with the same name, but at different levels or quality and quantity;
- When two fleets with the same name are entered in the Tyche input file (.tyi), such as through an external text editor like Windows Notepad), an error handling procedure identifies this and does not allow the .tyi to be opened. Duplicate names are not allowed, as they cause errors in collection objects;
- Amendment of the Multiple Level Constraints window so that an Asset Type's Default Level no longer appears in list of Levels to include in a Constraint;
- The error when deleting a Multiple Level Constraint using the delete or backspace key has been eliminated;
- The save file dialog box now pops up on top of all windows when closing Tyche;
- Inclusion of the risk and statistical error analysis into the data analysis performed by Tyche; and,
- Upon completion of the data analysis, WinZip is called via a command line interface to zip up the Tyche output (.tyo) file. If successful, the .tyo is then deleted to minimize usage of hard drive space.

2.2 Global error handling and logging

Within the programming language, Visual Basic 6.0, a number of ways to handle errors when they occur are offered. A brief summary is presented of what was used previously in Tyche, and possible alternatives for upgrade are debated. The final method for error handling in Tyche 2.3 is then discussed in detail.

[1] Functionality developed by Cheryl Eisler.

2.2.1 Coding options

Visual Basic 6.0 provides a number of procedures that a programmer can use to circumvent known behaviours or implement error handling functions:

a. **No Custom Error Handling (Default) -** if employed, Visual Basic simply stops execution of the program, displays an error message box, and gives an option to the user to *Debug* (if Visual Basic is installed) or *End* the program.

   This is the worst possible option and should be avoided in any but the most basic program.

b. **On Error Goto 0 -** disables any active error handling. Any errors raised will cause the Visual Basic error message dialog box to be displayed, after which the program will terminate.

   This is the default setting, which has been heavily utilised in previous versions of Tyche. Such use of error handling is not the best approach, as any error encountered will cause the application to quit abruptly and lose all unsaved data. The user has no ability to determine where the error occurred, or to recover from it.

c. **On Error Goto Line Label -** tells an application to immediately go to a specific line label when an error is detected. Code execution is continued from the line label. Once declared, the above statement enforces an error handling routine defined under “label”. This error handling will apply to the current procedure/function as well as to any other procedures called from within the current one (where this statement is declared) that have no error handling statements of their own (until the procedure itself returns control to its calling procedure).

   This can form the initial call to a custom error handling procedure. It allows the programmer to assign specific behaviours to known error codes, and allows the program to raise error messages with more detailed explanations of the problem. This was used only sporadically prior to Tyche 2.3, but is highly recommended for use.

d. **On Error Resume Next -** tells the application to ignore an error and simply continue execution on the next statement. This should be used with extreme caution, as its use can cause catastrophic data corruption or loss with no warning to the user.

   Tyche 2.2 (and lower) used this call on occasion when errors were known to occur with no significant consequences. Instead, the procedure in (b) should be used to trap and resolve the error.

e. **Error Handling Procedures -** functions can be designed specifically to trap and either report or resolve errors. Messages to the user can be tailored to report where the error occurred and known error codes can be diverted into appropriate procedures to resolve the problem (often with input data). Execution of the program can continue normally.

   Tyche 2.3 will utilize options (b) and (d) to deal with errors in the code.

2.2.2 Error handling techniques

It is considered good practice to allow an application to recover gracefully from an error rather than simply displaying the default error message and exiting the program. If possible, the user should be unaware an error has occurred and code should be written around the error to correct the problem. Often this is not possible, so some sort of message is required to be displayed to the user. Generally only an error message from the GUI should be displayed. A number of alternative design methods can be employed:

a. **Allow Parent Procedures to Handle Errors -** simply include an error handling declaration in parent procedure and allow it to handle any errors occurring in any called procedures. Generally, this is only recommended for simple applications as it is hard to predict the procedure execution order to arrive at the error.

   In previous versions of Tyche, this was used in the GUI to trap when the “Cancel” button was pressed when using the common dialog control to open files in Windows. Otherwise, the standard error message would appear. It was simple and got the job done, but was not user friendly.

b. **Include an Error Parameter in Custom Procedures -** by declaring an error parameter in each custom procedure, any error that occurs can be passed back to the calling function. Typically this value will be a Boolean, Long or String value.

   This gives the calling routine control of the error, but will not allow the program to resolve the issue within the problem function.

c. **Use Only Functions (No Subs) and Make the Return Value the Error Code -** making all procedures (subs) into functions and ensuring that all return values are error codes that can be tracked and passed back to the calling routine. Typically either Boolean (True for successful completion) or Long (0 for successful completion) values are used.

   Option (b) and (c) are functionally identical.

d. **Include an Error Property in Custom Classes -** whenever a class is created, include an error property which can be set whenever an error occurs. After any method/property/event in the class has been executed this property can be checked to see if an error has occurred. Typically this would be a Boolean value, although there may be additional properties for error numbers/descriptions.

   Because Tyche relies heavily on class manipulation, this could bog down the code unnecessarily. Proper form design and data declaration/manipulation obviates the need for this solution.

e. **Create an Error Handling Component -** this is the most robust way of handling errors, but also the one involving the most coding. This involves a small amount of code to create the component itself, but involves a lot of code to include the use of the error component within the code. Typically, this could be an enhancement to the previous idea by making the error property a class rather than a single value.

To give the programmer the ability to recover from specific error within a function, no error classes should be developed for Tyche. Instead, an error handling component should be added to every procedure/function.

f. **Raising Custom Errors -** custom errors can also be raised. Such method is employed if there is a need to perform some validation on user input and generate an error when the input did not meet validation criteria. By raising this as a custom error, the error handling routine would be allowed to handle the error.[2]

   This is often done when common errors reoccur in a piece of code to provide the program with a way of identifying the error and reporting it. It is better practice to design code such that repeated errors can not occur in the first place.

The general recommendation is to add error handling to every procedure, in any but the most trivial of applications. It does add an overhead to the length of written code and the time taken to code; however, templates can reduce the time taken to add the code, and the size added to the code is irrelevant compared with the benefits of having robust error handling.

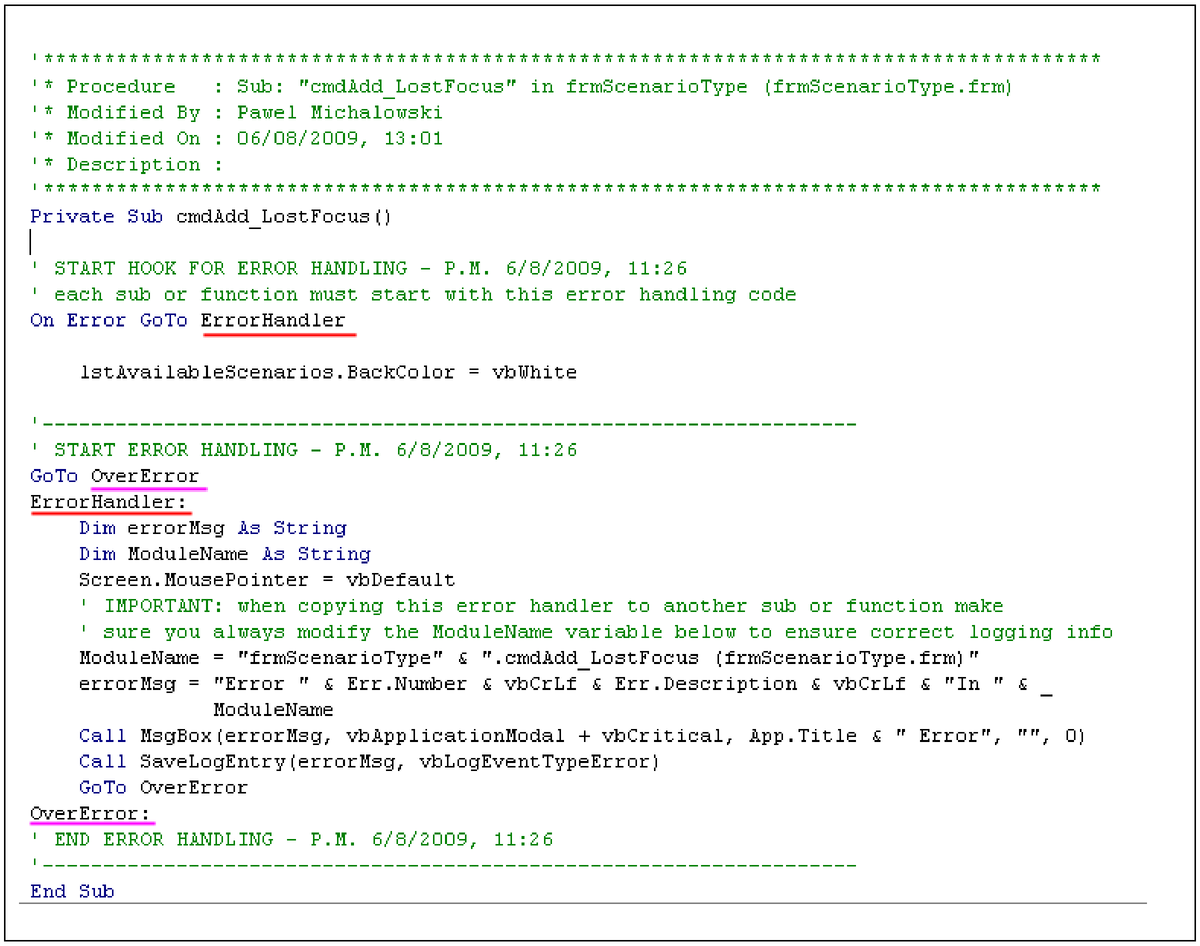
2.2.3 Error handling in Tyche 2.3

Tyche 2.3 employs a custom handling component, as illustrated in the example in Figure 1. The Tyche code also includes the writing of errors to an XML file, as well as the Windows' Events log. The log can be accessed through the Windows Event Viewer, either from Tyche under the *View* menu by selecting *Windows Event Viewer* (as illustrated in Figure 2) or through the *Start* menu -> *Control Pane*l -> *Administrative Tools* -> *Event Viewer*.

A sample log is shown in Figure 3, with an example error reported by Tyche in Figure 4. Like the Dashboard Debugger, the Windows Event Viewer is read-only. It provides a powerful tool for tracking errors in the simulation code, allowing the user to pinpoint the problem more quickly.

Different situations may require different solutions - Tyche utilises the one which was most appropriate for this kind of application.

[2] By convention, numbers from the constant vbObjectError + 1 and above can be used for custom errors.



```
'***************************************************************************************
'* Procedure   : Sub: "cmdAdd_LostFocus" in frmScenarioType (frmScenarioType.frm)
'* Modified By : Pawel Michalowski
'* Modified On : 06/08/2009, 13:01
'* Description :
'***************************************************************************************
Private Sub cmdAdd_LostFocus()

' START HOOK FOR ERROR HANDLING - P.M. 6/8/2009, 11:26
' each sub or function must start with this error handling code
On Error GoTo ErrorHandler

    lstAvailableScenarios.BackColor = vbWhite

'-------------------------------------------------------------------
' START ERROR HANDLING - P.M. 6/8/2009, 11:26
GoTo OverError
ErrorHandler:
    Dim errorMsg As String
    Dim ModuleName As String
    Screen.MousePointer = vbDefault
    ' IMPORTANT: when copying this error handler to another sub or function make
    ' sure you always modify the ModuleName variable below to ensure correct logging info
    ModuleName = "frmScenarioType" & ".cmdAdd_LostFocus (frmScenarioType.frm)"
    errorMsg = "Error " & Err.Number & vbCrLf & Err.Description & vbCrLf & "In " & _
               ModuleName
    Call MsgBox(errorMsg, vbApplicationModal + vbCritical, App.Title & " Error", "", 0)
    Call SaveLogEntry(errorMsg, vbLogEventTypeError)
    GoTo OverError
OverError:
' END ERROR HANDLING - P.M. 6/8/2009, 11:26
'-------------------------------------------------------------------
End Sub
```

Figure 1: Custom error handling as implemented in Tyche 2.3.

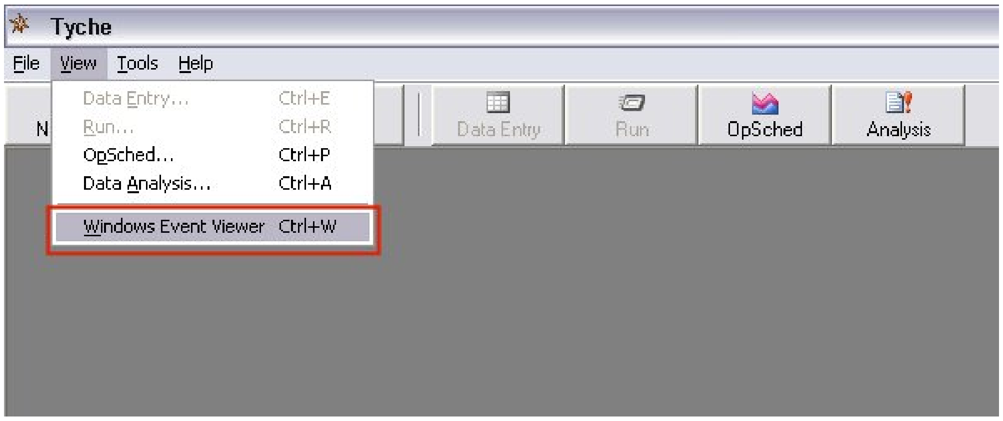
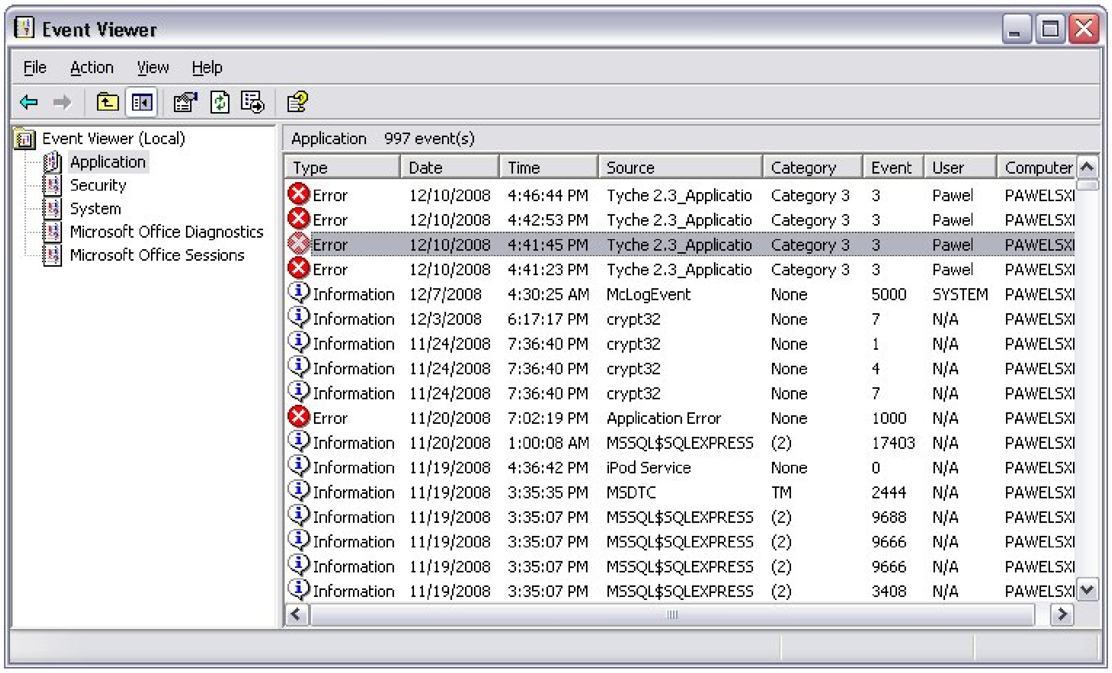


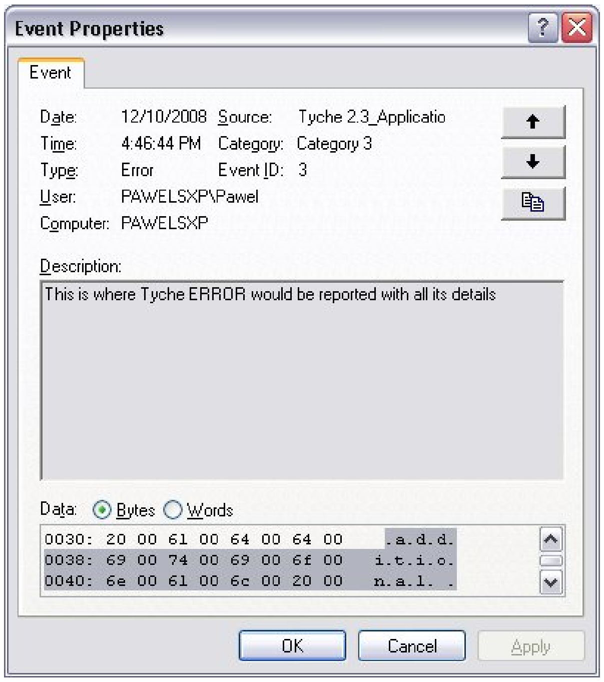
Figure 2: Opening Windows Event Viewer from Tyche 2.3.



Application 997 event(s)

Type	Date	Time	Source	Category	Event	User	Computer
Error	12/10/2008	4:46:44 PM	Tyche 2.3_Applicatio	Category 3	3	Pawel	PAWELSXI
Error	12/10/2008	4:42:53 PM	Tyche 2.3_Applicatio	Category 3	3	Pawel	PAWELSXI
Error	12/10/2008	4:41:45 PM	Tyche 2.3_Applicatio	Category 3	3	Pawel	PAWELSXI
Error	12/10/2008	4:41:23 PM	Tyche 2.3_Applicatio	Category 3	3	Pawel	PAWELSXI
Information	12/7/2008	4:30:25 AM	McLogEvent	None	5000	SYSTEM	PAWELSXI
Information	12/3/2008	6:17:17 PM	crypt32	None	7	N/A	PAWELSXI
Information	11/24/2008	7:36:40 PM	crypt32	None	1	N/A	PAWELSXI
Information	11/24/2008	7:36:40 PM	crypt32	None	4	N/A	PAWELSXI
Information	11/24/2008	7:36:40 PM	crypt32	None	7	N/A	PAWELSXI
Error	11/20/2008	7:02:19 PM	Application Error	None	1000	N/A	PAWELSXI
Information	11/20/2008	1:00:08 AM	MSSQL$SQLEXPRESS	(2)	17403	N/A	PAWELSXI
Information	11/19/2008	4:36:42 PM	iPod Service	None	0	N/A	PAWELSXI
Information	11/19/2008	3:35:35 PM	MSDTC	TM	2444	N/A	PAWELSXI
Information	11/19/2008	3:35:07 PM	MSSQL$SQLEXPRESS	(2)	9688	N/A	PAWELSXI
Information	11/19/2008	3:35:07 PM	MSSQL$SQLEXPRESS	(2)	9666	N/A	PAWELSXI
Information	11/19/2008	3:35:07 PM	MSSQL$SQLEXPRESS	(2)	9666	N/A	PAWELSXI
Information	11/19/2008	3:35:07 PM	MSSQL$SQLEXPRESS	(2)	3408	N/A	PAWELSXI

Figure 3: Sample of Windows Event Viewer.



Date: 12/10/2008 Source: Tyche 2.3_Applicatio
Time: 4:46:44 PM Category: Category 3
Type: Error Event ID: 3
User: PAWELSXP\Pawel
Computer: PAWELSXP

Description:
This is where Tyche ERROR would be reported with all its details

Data: [✓ Bytes] Words

```
0030: 20 00 61 00 64 00 64 00    .a.d.d.
0038: 69 00 74 00 69 00 6f 00   i.t.i.o.
0040: 6e 00 61 00 6c 00 20 00   n.a.l. .
```

Figure 4: Example of an error reported by Tyche.

2.2.4 Difficulties with the Windows Event Viewer

The Windows Event Viewer has a maximum size limit and overwriting restrictions for logging errors in its own files. If the maximum file size has been reached and the option to "overwrite as necessary" option has not been selected, the unspecified runtime error -2147024882 (System Error &H80004005 (-2147467259) will appear when trying to conduct an automated simulation. To correct the problem, got to *Start* menu -> *Control Pane*l -> *Administrative Tools* -> *Event Viewer*. Right-click on *Application* and select *Properties*. On the "General" tab, select either "Clear log" or "Overwrite events as needed". Alternatively, maximum file size can also be increased. More information is available at Ref. [6].

2.3 Integrated help support

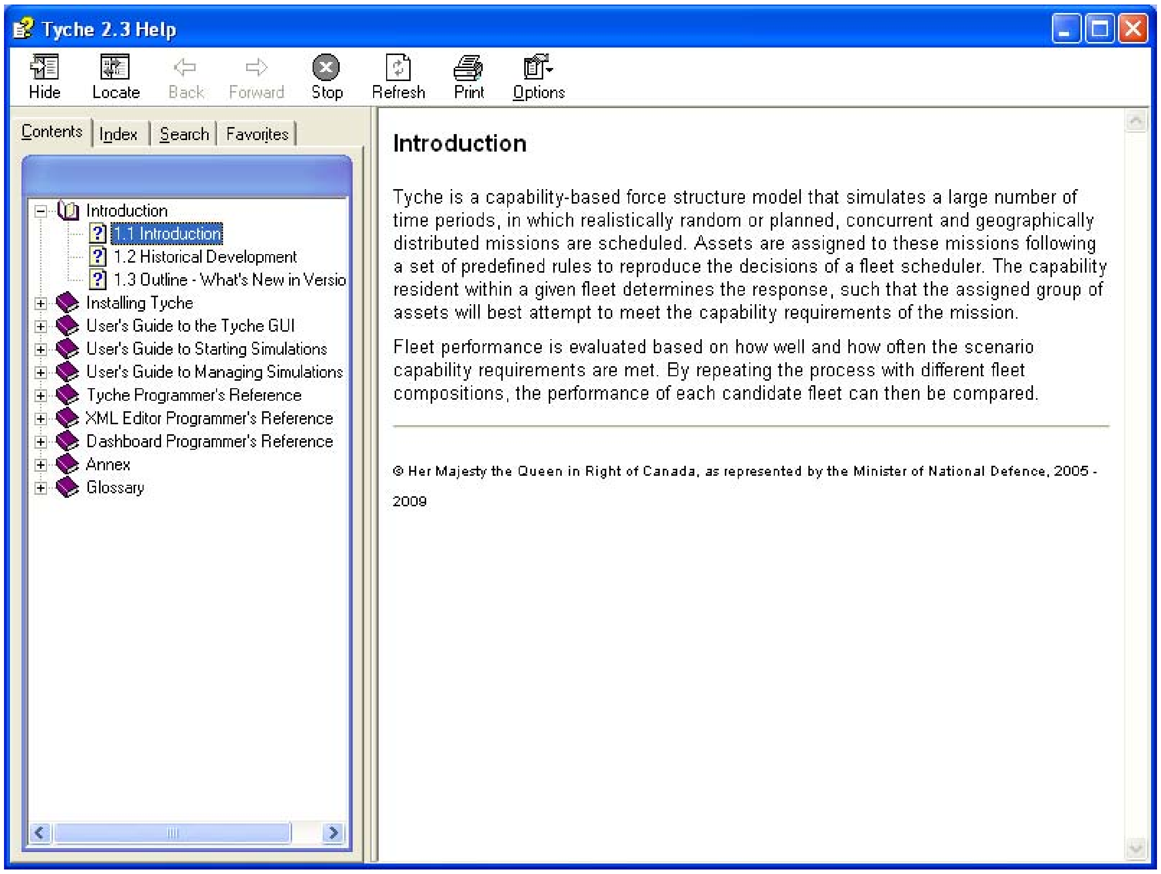
Since the Tyche applications are written in Visual Basic 6, there are only a few options available for integrating help support. These include ToolTip Text, "What's This?" functions, linked help applications, and context sensitive identification (ID). ToolTip Text has already been implemented for the controls in the Tyche GUI. "What's This" functions were ruled out in favour of context sensitive ID's (different functionality provided by the same form property). A compiled HTML help (CHM) file is linked to the Tyche executables and is used to provide context-sensitive help.[3]

Help files were written in HTML and compiled into a single CHM using Adobe RoboHelp. In the future, the HTML files can be used to support the development of a WIKI, as recommended in reference [7], if sufficient user interest can be generated and security issues overcome. For now, the CHM files are included with the executables, and can be accessed through the Tyche *Help* menu using *Contents…*, *Index…* or *Search…* to bring up a separate help window. An example of the help is shown in Figure 5.

Context sensitive ID's allow the user to open up a help topic specifically designed for the form that they are currently interacting with in an application. Programmatically, the HelpContextID property of a form can be set to the a numeric ID associated with the topic to display. However, it is more intuitive to reference the actual HTML file containing the topic. In order to do so, HTML support classes and functions have been created in the following files:

- clsHtmlHelp.cls
- clsFormHtmlHelpHandler.cls
- modHTMLHelp.bas

[3] In order to accomplish the connection, the help file to be used has to be specified by setting the App.HelpFile property in the code (e.g., App.HelpFile = App.Path & "\ Tyche.chm").


Tyche 2.3 Help
Hide
Locate
Back
Forward
Stop
Refresh
Print
Options
Contents
Index
Search
Favorites
Introduction
1.1 Introduction
1.2 Historical Development
1.3 Outline - What's New in Versio
Installing Tyche
User's Guide to the Tyche GUI
User's Guide to Starting Simulations
User's Guide to Managing Simulations
Tyche Programmer's Reference
XML Editor Programmer's Reference
Dashboard Programmer's Reference
Annex
Glossary

Introduction

Tyche is a capability-based force structure model that simulates a large number of time periods, in which realistically random or planned, concurrent and geographically distributed missions are scheduled. Assets are assigned to these missions following a set of predefined rules to reproduce the decisions of a fleet scheduler. The capability resident within a given fleet determines the response, such that the assigned group of assets will best attempt to meet the capability requirements of the mission.

Fleet performance is evaluated based on how well and how often the scenario capability requirements are met. By repeating the process with different fleet compositions, the performance of each candidate fleet can then be compared.

© Her Majesty the Queen in Right of Canada, as represented by the Minister of National Defence, 2005 - 2009

Figure 5: Example of the help files in Tyche.

This greatly simplifies the process of launching the HTML help in the right location, and is achieved by making the following call in the code:

```
Call HTMLHelp.OpenTopic("6_3_1_About_Box.htm", App.HelpFile,
frmAbout.hwnd)
```

The first parameter is the actual file name of the uncompressed HTML file containing the topic of interest. The second parameter is the name of the compiled HTML file, which in this case is referred to via a global application variable set at the beginning of the application.[4] When the user presses the [F1] key while a form is active, the topic linked to that form will then be displayed.[5] Such an approach is much more intuitive, and easier for future developers to change when additions are made to either the application or the help files.

[4] App.HelpFile, set in Private Sub MDIForm_Load() of frmParent.frm.

[5] To make a help window appear when the user presses the [F1] key, each form's WhatsThisHelp property must be set to False.

2.4 Operational Schedule search function

Tyche output files are extremely large (often > 256 MB) and, as a result, are slow or impossible to open using conventional text editors. However, the user may need to search through the output using keywords to identify places in the simulation where specific events are occurring. For example, a user may want to determine why no Assets are being assigned to a given Scenario. By searching for this event in the output, the user can determine if other events occurring at the same time may be interfering with Asset assignment. The OpSched Viewer was meant as a graphical representation of the output and did not allow the user to search the output directly.

A search function, similar to that of the "Find" function available in Notepad, under the "Edit" menu, has now been implemented. However, the search criteria are constructed from a set of embedded controls added to the OpSched Viewer, shown at the bottom of the screen capture in Figure 6. The criteria are constructed in common combinations that a user would use to debug the output.

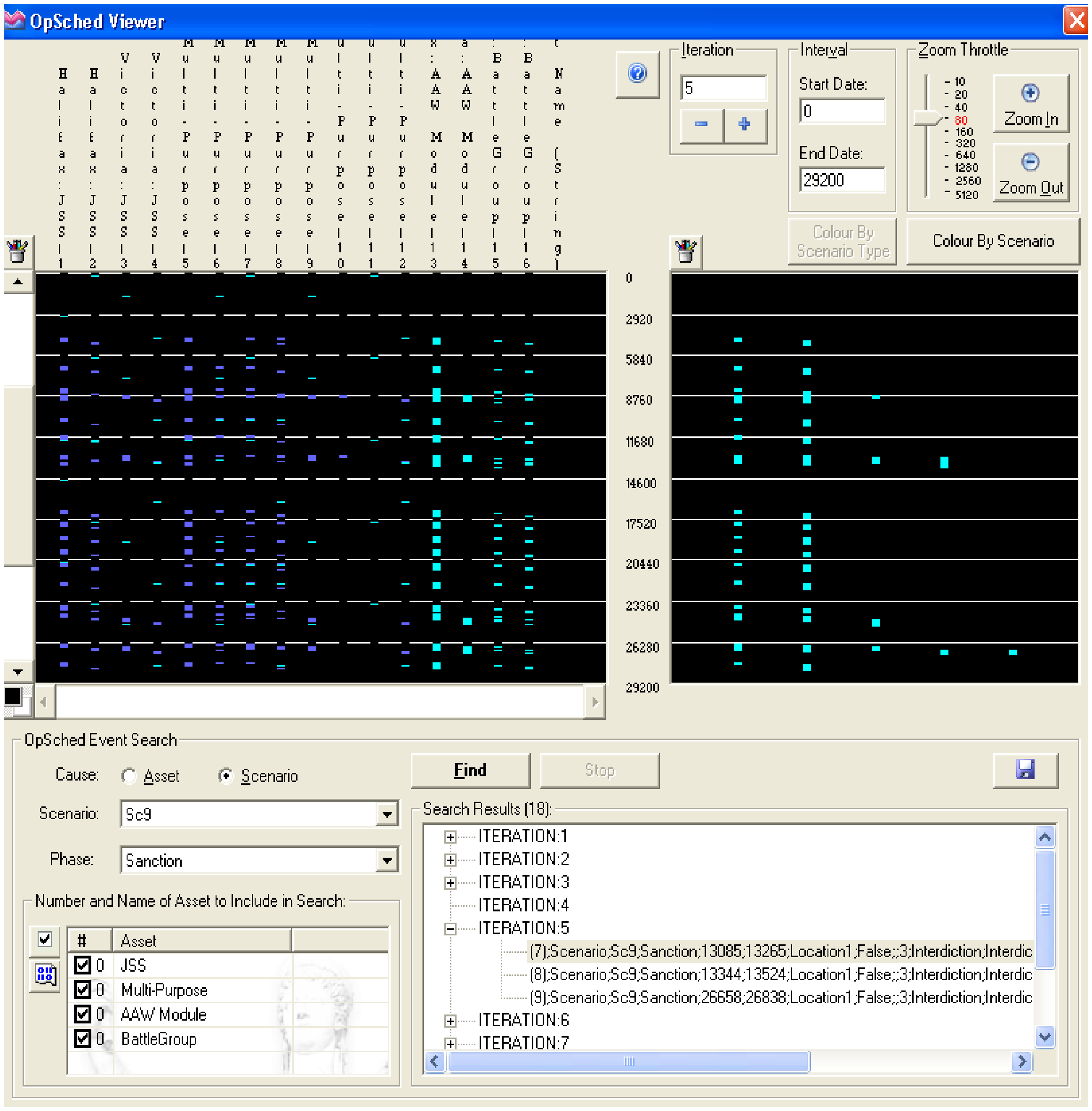
The search function looks for all instances in the Tyche output file where all requested objects are listed for the same event (all on one line of the output file). Objects in these strings are used to determine if the search criteria have been met. The .tyo output file has the following format for events in a simulation:

```
Asset;Asset Name;Level;Start Date;End Date;Base;Rescheduled
Event;Assigned Asset Name;Level;Base Departure Date;Theatre
Arrival Date;Theatre Departure Date;Base Arrival Date;Asset
Was Bumped;Number of Capabilities Missing at Required
Level;Capability Names (if any);Number of Capabilities
Missing at Marginal Level;Capability Names
```

Or:

```
Scenario;Name;Phase;Start Date;End Date;Theatre;Rescheduled
Event;Assigned Asset Name;Level;Base Departure Date;Theatre
Arrival Date;Theatre Departure Date;Base Arrival Date;Asset Was
Bumped;Number of Capabilities Missing at Required Level;Capability
Names (if any);Number of Capabilities Missing at Marginal
Level;Capability Names (if any)
```

Where the first object (delimited by semi-colons) is the Cause of the event (either Asset or Scenario), and the second item is the event Source (i.e., the name of the Asset or Scenario and the name or the corresponding Level or Phase).



Iteration: 5

Interval: Start Date: 0 End Date: 29200

Cause: ( ) Asset [✓ Scenario]

Scenario: Sc9

Phase: Sanction

Number and Name of Asset to Include in Search:

#	Asset
☑ 0	JSS
☑ 0	Multi-Purpose
☑ 0	AAW Module
☑ 0	BattleGroup

Search Results (18):

```
+ ITERATION:1
+ ITERATION:2
+ ITERATION:3
  ITERATION:4
- ITERATION:5
      (7);Scenario;Sc9;Sanction;13085;13265;Location1;False;;3;Interdiction;Interdic
      (8);Scenario;Sc9;Sanction;13344;13524;Location1;False;;3;Interdiction;Interdic
      (9);Scenario;Sc9;Sanction;26658;26838;Location1;False;;3;Interdiction;Interdic
+ ITERATION:6
+ ITERATION:7
```

Figure 6: The Operational Schedule Viewer.

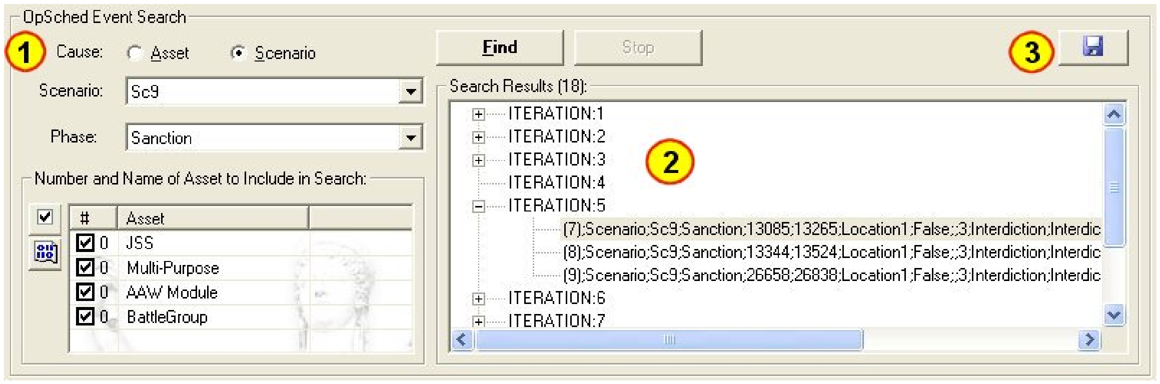
The search function is available to the user at all times. Radio buttons (Note 1 in Figure 7) allow the user to select the Cause of the event. Depending on the value of the radio button selected, either Asset or Scenario, the text headers and corresponding combo boxes will change. If the Cause of the event is an Asset, then the first combo box will be a choice of available Asset Types, based on the fleet in use for the .tyo file. The second combo box will display the possible Levels once an Asset Type has been selected. If the Cause of the event is a Scenario, the first combo box will show a list of all possible Scenarios. The second box will display possible Phases once a Scenario has been selected. The user can then select a Phase (optional), and zero or more assets that were assigned to the Scenario (optional).

When the Cause selected is a Scenario, a list view control with check-boxes will automatically load a list of Asset Types, and, for each, provide a box in which to enter a number to be selected in the search. There is also the option to specify if no Assets were assigned to the Scenario versus if any Type and number of Assets were assigned to the Scenario.

There are buttons to start (“Find”) a search and to stop (“Stop”) the search. Once initiated, the search looks for all matching instances of the search criteria in the .tyo file on a line-by-line basis. To make sure that search results are available to the user as soon as results are found (as the full search may take some time on large output files), the search function automatically updates the tree view containing the search results (Note 2 in Figure 7).

There is also a button that allows the user to save (“Save”) the search results to a text file (Note 3 in Figure 7).

A sample search is shown in Figure 7 for a Scenario where no assets have been assigned. The example utilized here is drawn from reference [1]. The following sub-sections will illustrate how a search is constructed and how to use the results.



Cause: Asset [✓ Scenario]

Scenario: Sc9

Phase: Sanction

#	Asset
✓ 0	JSS
✓ 0	Multi-Purpose
✓ 0	AAW Module
✓ 0	BattleGroup

Search Results (18):

```
ITERATION:1
ITERATION:2
ITERATION:3
ITERATION:4
ITERATION:5
    (7);Scenario;Sc9;Sanction;13085;13265;Location1;False;;3;Interdiction;Interdic
    (8);Scenario;Sc9;Sanction;13344;13524;Location1;False;;3;Interdiction;Interdic
    (9);Scenario;Sc9;Sanction;26658;26838;Location1;False;;3;Interdiction;Interdic
ITERATION:6
ITERATION:7
```

Figure 7: The Operational Schedule Viewer search function.

2.4.1 Search criteria

As illustrated under Note 1 in Figure 7, the event search criteria are constructed using a combination of objects. Depending on the Cause of the event that the user selects (Asset or Scenario), they will have the option of selecting from a list of Asset Types and Levels or Scenarios and Phases. Shown here is an example for Scenario *Sc9*, Phase *Sanction*.

In the Scenario search mode, the “Number and Name of Asset to Include in Search” option is activated. Here, the user can include a number of Asset Types in the search by clicking the check box beside the required Asset Type, as well as specifying the number of Assets of that Type to find assigned to the required Scenario. If the item is checked (INCLUDED in search) but the value remains 0 that tells the search function to find scenarios which <u>do not</u> have that Asset Type assigned. If all check boxes are toggled ON and all values are 0, then the search function looks for scenarios that have no assets assigned at all. If checkboxes are toggled OFF, then the search is performed ignoring these inputs.

There are two buttons next to the list view. The first toggles all check-boxes ON or OFF (INCLUDE or EXCLUDE from search); the second resets all items in the list to 0. This allows the user to quickly create common searches.

During Asset search mode, the “Number and Name of Asset to Include in Search” option remains disabled.

2.4.2 Search results

Once the user selects search criteria and clicks the Find button, the search automatically starts to populate the tree view (Note 2 in Figure 7) with matching search results. Each item found is displayed under the corresponding “ITERATION” parent node for easier navigation. When the user clicks on a search item found, the OpSched Viewer will automatically update its view to display the appropriate iteration in the graphical representation above the search.

If a search is taking a long time, the user can choose to terminate it by pressing the Stop button. If a search is terminated before reaching the end of the file, the user can not resume from where they stopped. Pressing the Find button will begin the search again from the beginning of the .tyo file.

2.4.3 Save search results to file

Once a search is completed or stopped, the user can save the search results to a text file. Tyche attempts to save the search results to the default location of the executable in a file named “OpSchedSearchResults.txt” by prompting the user with a Save As dialog box. Using the Save As dialog, the user can choose any destination and/or file name to save the search results. Aside from the search results, the file also contains useful information such as title, date and time of the search performed. The format of the search output text file is shown in Figure 8.

```
TITLE: Tyche 2.3 OpSched Search Results
DATE: 6/4/2009
TIME: 5:57:09 PM

SEARCH DATA:

ITERATION:1
   (1);Scenario; ....
ITERATION:2
   (2);Scenario;Sc9...
   (3);Scenario;Sc9;Sanction...
```

Figure 8: The search function text file output format.

2.5 Separation of the Simulation Engine from the Tyche GUI

Prior to version 2.2 of Tyche, there was only one Tyche application/executable. Tyche 2.2 introduced the XML Editor and Dashboard applications in addition to the Tyche Editor. The XML Editor facilitates simulation setup parameters, which are then passed to the Dashboard to manage the queue of simulations running on a computer. The Tyche Editor was used for data input, simulations (running with the GUI suppressed), and operational schedule viewing.

In Tyche 2.3, four applications make up the suite: GUI, Simulation Engine (SE), Dashboard and XML Editor. The main focus in the redesign of Tyche 2.3 is the offloading of simulation procedures from the main Tyche Editor. In version 2.3, the Tyche Editor ("Tyche.exe") is split into the Tyche GUI ("Tyche-gui.exe") and the Tyche SE ("Tyche-se.exe"). The Tyche GUI (Tyche-gui.exe) continues to be the primary application in which users may open, modify and save input files. The Tyche GUI also contains the OpSched (operational schedule) viewer, which can be used to view the results of individual simulation iterations. Simulations can still be conducted from the GUI; the difference now lies in the program called to conduct the simulation.

There are two options available in the GUI to run a simulation. The default setting is to submit jobs through the XML Editor to the Dashboard. The Dashboard then calls the SE from the command line to conduct the simulation as a silent background process. It manages communication with one or more instances of the SE and the Windows Registry, and reports progress back to the user. Alternatively, the user can select the "Run in Debug Mode" option under the Tools menu in the GUI. This runs a simulation directly from the GUI, without calling the Dashboard or the SE.[6] There are two advantages to running a simulation in debug mode. The first is when the user wants to run test simulations (typically 1-5 iterations) for immediate viewing in the OpSched Viewer. The user can save time and effort when tracing back and forth between output and input data, and making and testing small changes. Secondly, because the SE is called as an independent process, running in debug mode from the Microsoft Visual Basic 6.0 development environment is the only way a user/programmer can debug the code used to run the simulations.

Functionally, the relationships between each of the applications in the Tyche suite are illustrated in Figure 9. Components that span vertical bands of the flowchart are elements of code shared between modules. The interested reader is referred to Ref. [3] or the integrated help file for a functional description of each application in the Tyche suite.

Figure 10 documents all individual and shared forms, modules, and classes in the Tyche application suite. This sharing (shown by boxes overlapping applications) avoids duplication of code, and the associated potential for error in failing to update multiple code sources. The simulation engine takes significant advantage of code sharing to utilize only components necessary to run a simulation. As a result, the SE is smaller (no graphical interface) and cleaner (rather than forcing suppression of the Editor from the Dashboard in Tyche 2.2.).

The command line functionality of the GUI and SE has been tailored to each of their new roles. The GUI can be launched from the command line with optional arguments for .tyi or .tyo files. Specifying a .tyi file will open the Data Entry Environment with the identified input file, whereas

[6] Using the Run form resurrected from Tyche version 2.0 [1].

a .tyo file will trigger the OpSched Viewer on the output file. As for the SE, the file name and type are not optional, but it has different behaviours and additional arguments depending on the file type. For the Tyche SE, the expected and valid types are either:

- .xml - Tyche starts silently in the background in automation mode. When an XML file is received on the command line, the data is parsed using the XML parser in the Simulation Instance module. The .tyi file referenced within the XML file is then opened with the Simulation Engine.
- .tyi with additional arguments in the form "[full file path and name].tyi;[fleet name];[number of iterations];[number of years];[random seed];[scenario types, separated by commas]" [7] - Tyche starts silently in the background in automation mode. The .tyi file is then opened with the Simulation Engine and the simulation variables are loaded based on the arguments given.

Creating a separate executable with just the core simulation elements that can be called from the command line restructures the application suite so that it can be used more easily and efficiently. This also applies to future users by positioning the application for automated use in a parallel processing environment.

[7] Fleets and scenario types must correspond exactly to the names associated in the .tyi file or an error will occur. The user can select one or more scenarios types to include in the simulation run. Any arguments with spaces or restricted characters must be given inside quotation marks for the optional arguments to work.

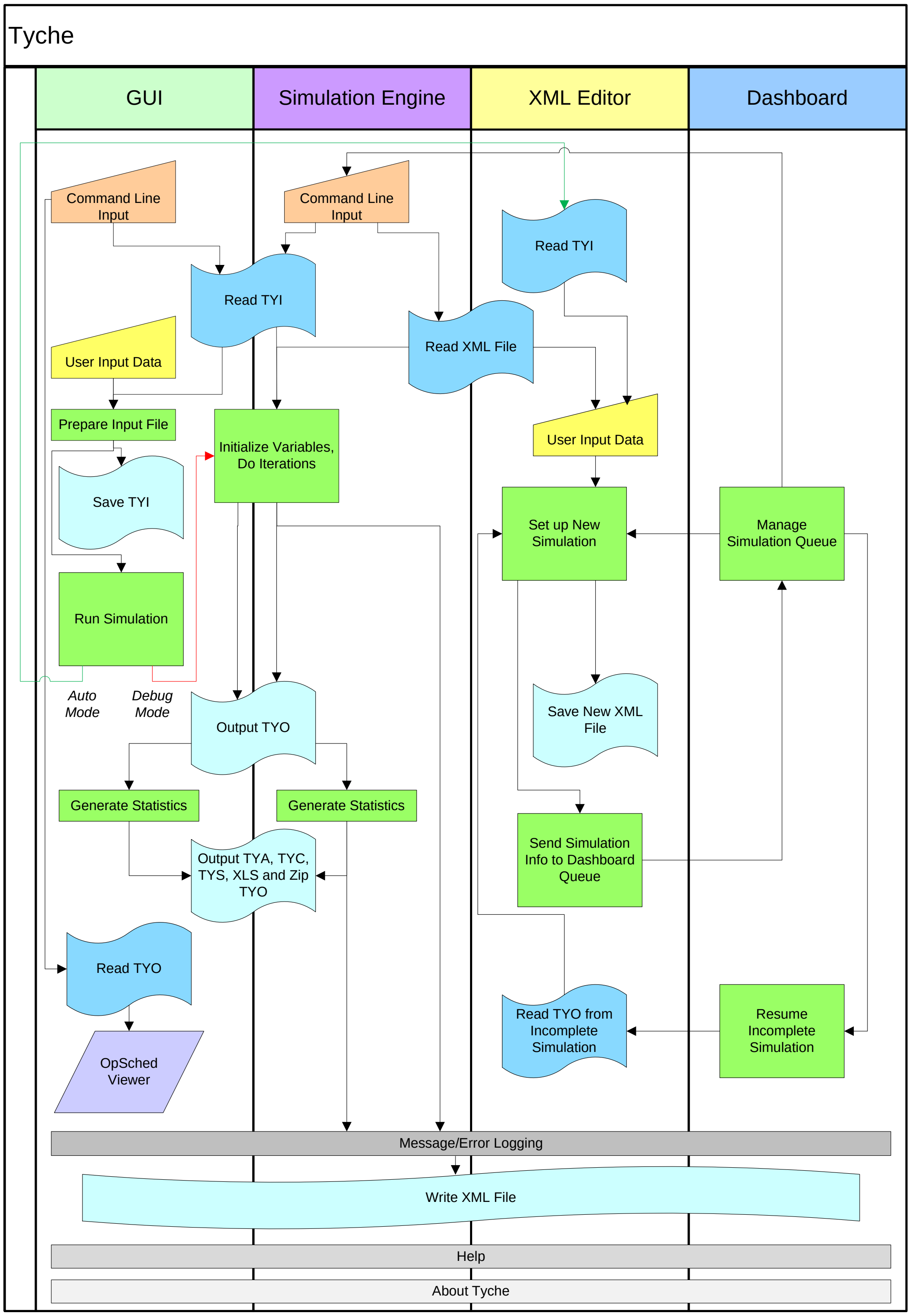


Figure 9: Functional flowchart of the Tyche application suite.

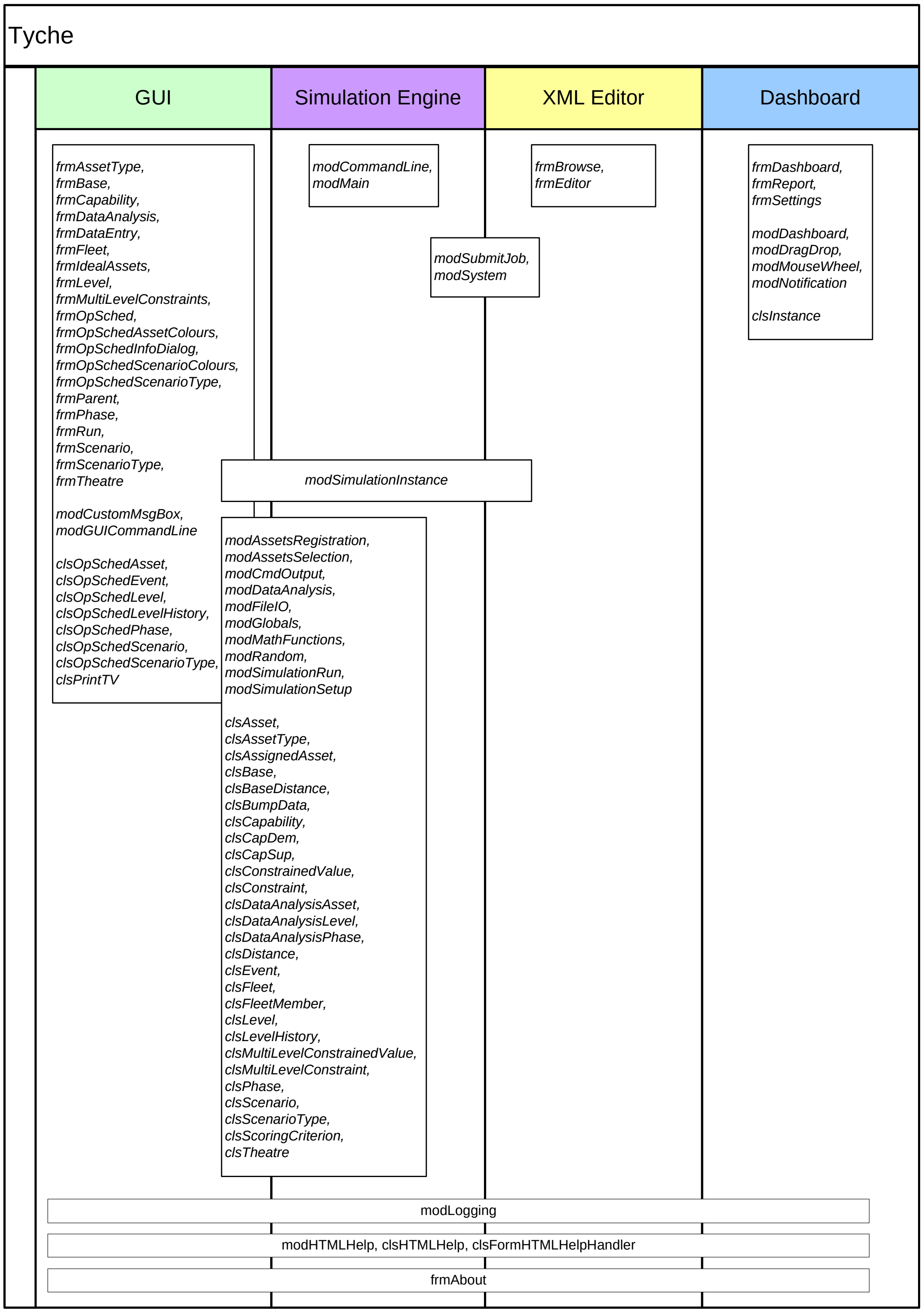

Tyche
GUI
Simulation Engine
XML Editor
Dashboard
frmAssetType, frmBase, frmCapability, frmDataAnalysis, frmDataEntry, frmFleet, frmIdealAssets, frmLevel, frmMultiLevelConstraints, frmOpSched, frmOpSchedAssetColours, frmOpSchedInfoDialog, frmOpSchedScenarioColours, frmOpSchedScenarioType, frmParent, frmPhase, frmRun, frmScenario, frmScenarioType, frmTheatre
modCustomMsgBox, modGUICommandLine
clsOpSchedAsset, clsOpSchedEvent, clsOpSchedLevel, clsOpSchedLevelHistory, clsOpSchedPhase, clsOpSchedScenario, clsOpSchedScenarioType, clsPrintTV
modCommandLine, modMain
frmBrowse, frmEditor
modSubmitJob, modSystem
frmDashboard, frmReport, frmSettings
modDashboard, modDragDrop, modMouseWheel, modNotification
clsInstance
modSimulationInstance
modAssetsRegistration, modAssetsSelection, modCmdOutput, modDataAnalysis, modFileIO, modGlobals, modMathFunctions, modRandom, modSimulationRun, modSimulationSetup
clsAsset, clsAssetType, clsAssignedAsset, clsBase, clsBaseDistance, clsBumpData, clsCapability, clsCapDem, clsCapSup, clsConstrainedValue, clsConstraint, clsDataAnalysisAsset, clsDataAnalysisLevel, clsDataAnalysisPhase, clsDistance, clsEvent, clsFleet, clsFleetMember, clsLevel, clsLevelHistory, clsMultiLevelConstrainedValue, clsMultiLevelConstraint, clsPhase, clsScenario, clsScenarioType, clsScoringCriterion, clsTheatre
modLogging
modHTMLHelp, clsHTMLHelp, clsFormHTMLHelpHandler
frmAbout

Figure 10: Individual and shared code within the Tyche application suite.

2.6 Additional information

Detailed information about Tyche 2.3 can be found in the Tyche HTML help file. Additional information about the developmental process is captured in the final (cumulative) weekly status report [8].

3 Future work

The primary development goal for Tyche is now to significantly improve the range of problems on which it can operate. By increasing the software's flexibility and speed of operation, the Maritime Operational Research Team can provide higher quality decision support to the Canadian navy in a shorter period of time, addressing more of the issues that are currently faced with limited resources [9].

The work completed to date prepares the software for further upgrades by splitting the simulation engine from the graphical user interface and improving error handling abilities. The next phase will consist of converting the simulation engine to a faster processing code and adapt the simulation engine to run efficiently in a parallel processing environment. Future phases will broaden the scope of problems the model can handle through:

- Developing a user-defined timescale to model assets with shorter usage patterns;
- User-defined asset assignment heuristics to allow greater flexibility and realism for risk mitigation through more sophisticated mission planning schemes;
- Improving file output formats to tailor data analysis and visualization; and,
- Redesigning the event handler to allow for concurrent operations, waypoints (e.g. forward deployment of assets), and additional locations for asset events (such as allowing critical maintenance to occur at the nearest base vs. home base) [9].

References

[1] Allen, D., Eisler, C., and Forget, A. (2006), A user guide to Tyche version 2.0: providing a joint flavour to Tyche, (DRDC CORA TR 2006-14) Defence R&D – Canada CORA.

[2] Eisler, C. (2007), Improving the ability to model future maritime force structures: from Tyche version 1.0 to 2.1, (DRDC CORA TM 2007-03), Defence R&D Canada – Centre for Operational Research and Analysis.

[3] Heppenstall, D. (2007), A User's Guide and Programmer's Manual to Tyche Version 2.2: Handling Simulations with Greater Efficiency and Flexibility, (DRDC CORA CR 2007-01), Defence R&D Canada – Centre for Operational Research and Analysis.

[4] Allen, D., Blakeney, D., Burton, R.M.H., Purcell, LCdr D. (2005), Fleet Mix Study: Determining the Capacity and Capability of the Future Naval Force Structure, (DRDC CORA TR 2005-38), Defence R&D Canada – Centre for Operational Research and Analysis.

[5] Bourque, A. and Eisler, C. (2009), Fleet Mix Study Iteration II: Making the Case for the Capacity of the "Navy After Next", (DRDC CORA TR DRAFT), Defence R&D Canada – Centre for Operational Research and Analysis.

[6] King, S. (21 January 2006), Visual Basic(Microsoft): Version 5 & 6 Forum App.LogEvent error -2147024882 (System Error &H80004005 (-2147467259) (Online), Tecumseh Group, Inc., http://www.tek-tips.com/viewthread.cfm?qid=1180118&page=1 (16 October 2009).

[7] Eisler, C. and Augusta, C. (2008), The Addition of Help Support to Tyche, (DRDC CORA TN 2008-034), Defence R&D Canada – Centre for Operational Research and Analysis.

[8] Michalowski, P. (2009), Weekly Status Report for Tyche 2.3 Project.

[9] Eisler, C. (2008), Force Structure Analysis Tool Development for Capability-Based Planning, Applied Research Project Description (Thrust 11ic), Defence R&D Canada – Centre for Operational Research and Analysis.

List of symbols/abbreviations/acronyms/initialisms

CORA	Centre for Operational Research and Analysis
CHM	Compiled Help File
DND	Department of National Defence
DRDC	Defence Research & Development Canada
FMS	Fleet Mix Study
GUI	Graphical User Interface
ID	Identification
MB	Megabytes
MORT	Maritime Operational Research Team
MVC	Model-View-Controller
OpSched	Operational Schedule
SE	Simulation Engine
.tyi	Tyche Input File
.tyo	Tyche Output File

Distribution list

Document No.: DRDC CORA CR 2009-005

	LIST PART 1: Internal Distribution by Centre
1	DRDC CORA/MORT (PDF)
2	DRDC CORA/MORT Attn: Cheryl Eisler (1 hard copy + PDF)
2	DRDC CORA/MORT Attn: Alex Bourque (1 hard copy + PDF)
2	DRDC CORA/Library (1 hard copy + PDF)
7	TOTAL LIST PART 1
	LIST PART 2: External Distribution by DRDKIM
1	DSTM (PDF)
1	DRDC/DRDKIM 3 (PDF)
2	TOTAL LIST PART 2
9	**TOTAL COPIES REQUIRED**

DOCUMENT CONTROL DATA

(Security classification of title, body of abstract and indexing annotation must be entered when the overall document is classified)

1. ORIGINATOR (The name and address of the organization preparing the document. Organizations for whom the document was prepared, e.g. Centre sponsoring a contractor's report, or tasking agency, are entered in section 8.) Pawel Michalowski Brainhunter (Ottawa) Inc. 1545 Carling Avenue Suite 600 Ottawa, ON Canada K1Z 8P9	2. SECURITY CLASSIFICATION (Overall security classification of the document including special warning terms if applicable.) UNCLASSIFIED

3. TITLE (The complete document title as indicated on the title page. Its classification should be indicated by the appropriate abbreviation (S, C or U) in parentheses after the title.)

Tyche Version 2.3: Supporting Documentation

4. AUTHORS (last name, followed by initials – ranks, titles, etc. not to be used)

Michalowski, P.

5. DATE OF PUBLICATION (Month and year of publication of document.) October 2009	6a. NO. OF PAGES (Total containing information, including Annexes, Appendices, etc.) 40	6b. NO. OF REFS (Total cited in document.) 9

7. DESCRIPTIVE NOTES (The category of the document, e.g. technical report, technical note or memorandum. If appropriate, enter the type of report, e.g. interim, progress, summary, annual or final. Give the inclusive dates when a specific reporting period is covered.)

Contract Report

8. SPONSORING ACTIVITY (The name of the department project office or laboratory sponsoring the research and development – include address.)

Defence R&D Canada
101 Colonel By Drive
Ottawa, Ontario K1A 0K2

9a. PROJECT OR GRANT NO. (If appropriate, the applicable research and development project or grant number under which the document was written. Please specify whether project or grant.) ARP 11ic	9b. CONTRACT NO. (If appropriate, the applicable number under which the document was written.) EN537-8-4015/182/ZL
10a. ORIGINATOR'S DOCUMENT NUMBER (The official document number by which the document is identified by the originating activity. This number must be unique to this document.) DRDC CORA CR 2009-005	10b. OTHER DOCUMENT NO(s). (Any other numbers which may be assigned this document either by the originator or by the sponsor.)

11. DOCUMENT AVAILABILITY (Any limitations on further dissemination of the document, other than those imposed by security classification.)

Unlimited

12. DOCUMENT ANNOUNCEMENT (Any limitation to the bibliographic announcement of this document. This will normally correspond to the Document Availability (11). However, where further distribution (beyond the audience specified in (11) is possible, a wider announcement audience may be selected.))

13. ABSTRACT (A brief and factual summary of the document. It may also appear elsewhere in the body of the document itself. It is highly desirable that the abstract of classified documents be unclassified. Each paragraph of the abstract shall begin with an indication of the security classification of the information in the paragraph (unless the document itself is unclassified) represented as (S), (C), (R), or (U). It is not necessary to include here abstracts in both official languages unless the text is bilingual.)

Tyche is an application that was developed by the Maritime Operational Research Team in 2004. It is a stochastic force structure model assists that simulating the demand placed on future maritime fleets. After creating a realistic set of missions, fleets and fleet assets are scheduled to respond to the given circumstances. However, Tyche simulations take time to set up and can be computationally quite expensive. To improve both speed and the user’s ability to administer and maintain a queue of simulations, a new Tyche Simulation Engine (Tyche SE) module has been derived from the existing Tyche application. By sharing the same code base while restructuring the architecture, a background process version of Tyche (Tyche SE) is used solely for the purpose of running automated simulations through the Tyche simulation management tool dashboard. To maintain backwards compatibility and debugging functions, simulations can still be run from the Tyche graphical user interface (GUI). The GUI now includes several new features and code improvements, including compiled help files.

14. KEYWORDS, DESCRIPTORS or IDENTIFIERS (Technically meaningful terms or short phrases that characterize a document and could be helpful in cataloguing the document. They should be selected so that no security classification is required. Identifiers, such as equipment model designation, trade name, military project code name, geographic location may also be included. If possible keywords should be selected from a published thesaurus, e.g. Thesaurus of Engineering and Scientific Terms (TEST) and that thesaurus identified. If it is not possible to select indexing terms which are Unclassified, the classification of each should be indicated as with the title.)

Tyche; Force Structure Analysis; Help Files; Model-View-Controller Architecture; Search Function;

Defence R&D Canada

Canada's leader in Defence
and National Security
Science and Technology

R & D pour la défense Canada

Chef de file au Canada en matière
de science et de technologie pour
la défense et la sécurité nationale

DEFENCE DÉFENSE

www.drdc-rddc.gc.ca