TYCHE 3.2 Upgrade

Chad Watson

Prepared by:

International Safety Research
38 Colonnade Road North
Ottawa, (ON)
Canada

PSPC Contract Number: W7714-156105
Technical Authority: Cheryl Eisler and Fred Ma
Contractor's date of publication: May 2018

Defence Research and Development Canada

Contract Report
DRDC-RDDC-2018-C095
May 2018

TYCHE 3.2 Upgrade
Contractor Report

W7714-156105 T039
ISR Report 6111-01-01
Version 6.0
22 May 2018

Presented to:
Cheryl Eisler and Fred Ma

Centre for Operational Research and Analysis
Defence Research and Development Canada
Government of Canada

Prepared by:

International Safety Research
38 Colonnade Road North
Ottawa, Ontario
Canada K2E 7J6

QUALITY ASSURANCE AND VERSION TRACKING

Authorization

Title	**TYCHE 3.2 Upgrade Contractor Report**	
Report number	6111-01-01	
Version	6.0	Signature
Prepared by	Chad Watson	Signed.
Reviewed by	Malika Bongue Boma	[illegible]
Approved by	Ian Becking	[illegible]
Approved for Corporate Release by	Mike McCall	[illegible]

Version Tracking

Ver.	Action	By	Date
1.0	Table of Contents released to Client	McCall	13 Dec 2017
2.0	Reverse sections 2 and 3, release to client	McCall	18 Dec 2017
3.0	Draft report. Release to client.	Becking for McCall	01 Feb 2018
4.0	Address comments. Release to client.	McCall	20 Mar 2018
5.0	Client corrections for publication.	Eisler	24 April 2018
6.0	Client corrections for publication.	Eisler	22 May 2018

Abstract

Tyche is a Monte Carlo discrete-event simulation application tool developed by Defence Research and Development Canada's Centre for Operational Research and Analysis (DRDC CORA) to support the Canadian Armed Forces in conducting capability-based planning for force structure analysis. As part of a phased upgrade approach to enable the tool to handle medium-to-large-scale, multi-objective optimization problems, DRDC CORA established a requirement to transition the Tyche application to a fully-supported, modern programming language and perform significant upgrades.

Tyche 3.2 is a natural extension to Tyche 3.1. In Tyche 3.1 flexibility was added to allow a user to select a time scale appropriate to the problem at hand, and the application was readied for high performance computing environments (HPC) through the implementation of fractional simulations. In Tyche 3.2, first-use defects of Tyche 3.1 were rectified to improve the software reliability, performance benchmarking was identified, and the documentation was given a refresh.

Résumé

Tyche est un outil de l’application de simulation à événements discrets de Monte Carlo développé par le Centre d’analyse et de recherche opérationnelle de Recherche et développement pour la défense Canada (CARO RDDC) afin d’aider les Forces armées canadiennes à effectuer la planification fondée sur les capacités pour l’analyse de la structure des forces. Dans le cadre d’une approche de modernisation graduelle permettant à l’outil de traiter des problèmes d’optimisation multiobjectif de moyenne à grande envergure, CARO RDDC a établi le besoin de transformer l’application Tyche en un langage de programmation moderne avec soutien complet et d’effectuer des mises à niveau importantes.

Tyche 3.2 est une suite logique de Tyche 3.1. On a ajouté de la souplesse à Tyche 3.1 pour permettre à un utilisateur de sélectionner l’échelle temporelle appropriée au problème à l’étude, et l’application a été préparée pour les environnements de calcul de haute performance (CHP) par la mise en place de simulations fractionnelles. Dans Tyche 3.2, les défauts à la première utilisation de Tyche 3.1 ont été corrigés pour améliorer la fiabilité du logiciel, des tests de performance ont été déterminés, et la documentation a été mise à jour.

TABLE OF CONTENTS

1. INTRODUCTION

Tyche is a Monte Carlo discrete-event simulation software tool that was developed by Defence Research and Development Canada - Centre for Operational Research and Analysis (DRDC CORA) to allow the Royal Canadian Navy (RCN) to conduct capability-based planning (CBP) for force structure analysis. In the interest of broadening the application of Tyche to a joint environment (i.e., across all services of Army, Navy and Air Force) and enabling the tool to handle medium-to-large-scale multi-objective optimization problems, Tyche was converted to a fully supported modern programming language (Microsoft Visual C#.NET).

Versions 3.0 and 3.1 of Tyche introduced major functional changes to Tyche. Version 3.2 of the software focused on fixing first-use defects found in relation to the previous major changes, as well as updated help documentation to reflect such changes.

Tyche 3.2 addresses version 3.1 bugs through extensive testing, and confirms that the parallel simulation processing time is close to specification. The parallel simulation testing procedure is provided for performance testing within the high-performance computing (HPC) environment.

1.1 Purpose

The purpose of this report is to provide an overview of the work performed in the upgrade to Tyche 3.2, discuss issues that arose, and present recommendations for furthering the development of Tyche.

At a high level, the following work was performed:

- Fixed bugs and implemented enhancements found in Tyche 3.1 (see section 2.2);
- Proposed a parallel simulation testing procedure within the HPC environment, and brought the parallel simulation processing time near to specification (See section 2.3); and
- Develop a serial and parallel processing test plan suitable for use within an HPC environment; and
- Updated the TYCHE help documentation [1].

1.2 Timeline

The Tyche 3.2 project commenced Dec 4, 2017. Development occurred over the months of December and January, with minor report writing updates occurring in February and March. The project was deemed complete by March 31, 2018.

2. TASK COMMENTARY

2.1 Overview

A brief overview of the outcome for each task performed (see Annex B.1 for the original SOW reference) is provided in this section.

2.2 Task 1 – Fix Bugs

Tyche version 3.2 includes several fixes and enhancements. The Tyche User guide section titled “Requirements for New Development” contains a listing of the notable fixes and enhancements. The information is reproduced here.

2.2.1 Fixes

1. **Erroneous Title Bar Path.** The title bar is used to maintain situational awareness on which .tyi (or .tyo) is loaded. In certain cases, the title bar was not being updated correctly.
2. **Simulation Editor Help Links.** The F1 and Help button response was not complete for the appropriate areas of context sensitive help. In some cases, the code level flag was not being set for the correct area of help. The code was re-factored to ensure the flags were being set, and, the labels were click enabled so that the cursor did not have to be in the control to trigger the help.
3. **Tyche Graphic User Interface (GUI) Not Saving Run Parameters**. When the GUI run dialogue prompted for a save of the .tyi, it was not saving the appropriate run parameters.
4. **Error Logging.** When an error occurred, the time-out error was not being reported in the appropriate extensible markup language (XML) file. This was fixed, and the logging capability was re-factored such that the errors are now output to a .log file instead of the .xml file.

2.2.2 Enhancements

1. **Version and Editions.** The documentation now utilizes a variable for the current software version that is used to update the text throughout the documentation. This reduces the publishing time when a new upgrade is released. The version number and release date are used to calculate an identifying documentation **Edition** number.
2. **Execution Type Saved to .tyi.** The .tyi used to only store three run parameters; however, a fourth that captured the intended execution type was required and implemented.
3. **Create HPC Job.** For ease of access, the functionality related to generating the HPC files for an HPC job was made available in the GUI’s *Run Simulation* form.

2.3 Task 2 – Propose and Implement Performance Test

This task was to develop a process to test the functionality of the HPC implementation of Tyche in both parallel and serial mode subject to the following constraints:

- The test plan must include the minimum number of steps to test out the parallel and serial functionality on the HPC system.
- The parallel version must produce the same output as the serial version (identical .tyo (zipped), .tya., .tyc, .tys., and .xls files) with no compiled or run time errors for the test input (.tyi) file(s) provided by the Technical Authority (TA).
- At full central processing unit (CPU) loading on *N* CPUs, *N* serial simulations should complete in *X* time (when measuring the total simulation run time, from the time the simulation log files record the simulation as starting to the time all iterations are recorded as completed).
- A single simulation of the parallel version at full CPU loading on *N* CPUs must not take longer to run than *(X/N)*1.1* (or no more than a 10% increase in total run time than the serial version).

A test plan was developed to characterize the performance of Tyche in serial vs. parallel mode. The tests were run on a contractor test server with the intent that the tests be repeated on a HPC system by the TA.

In addition, the performance scalability with increasing number of Cores was characterized, and the code was profiled to identify any other performance bottlenecks or areas of concern.

2.3.1 Definitions

- **CPU:** A physical CPU is a chip within its own socket located on the computer motherboard. It contains one or more processing cores.

- **Core:** Within a CPU chip there may be multiple CPU Cores. Each core contains a physical duplicate of the processing hardware of the CPU. All Cores exist within the same CPU chip and socket.

- **Logical Processors:** On hyperthreaded CPUs, each Core may appear to the Operating System as 2 Logical Processors. The physical CPU core hardware is shared between the logical processors.

  For example, on the Dual Intel XEON™ E5 2650 v2 2.6Ghz test server the system info shows:

  - Sockets: 2
  - Cores: 16
  - Logical Processors: 32

  In this case, there are 2 physical CPU chips, each containing 8 cores, each hyperthreaded. To the Windows® operating system (O/S), this appears as 32 logical processors.

- **Process**. An instance of an executing program (application, service). Each process has its own memory space and other resources. Each process contains one or more threads.

- **Thread**. A thread is the smallest set of one or more execution paths that can be managed independently. Threads share memory and resources with the parent process. A thread is scheduled and executed on a processor.

For Tyche, each iteration in parallel mode is executed within the main thread of a separately spawned simulation engine (SE) process. In contrast, when Tyche executes a simulation in serial mode, all iterations are executed within a single SE process.

2.3.2 Test Methodology

Exploratory testing was initially conducted on a laptop to get an understanding of Tyche performance characteristics, identify performance bottlenecks, develop the testing plan, and generate preliminary test data. Test results documented in this report were then performed on a dual CPU (16 core) Windows® server that is similar in architecture to a single server/blade of the target HPC system. Exploratory testing determined that the Tyche SE execution speed was primarily CPU bound during simulation iterations (excluding post- processing phase). Since input/output (I/O) was not a significant factor[1], elapsed time was used as the primary test metric.

Tests cases were developed to compare the performance of parallel vs. serial execution under equivalent CPU loading and with same inputs and number of iterations (defined as concurrency). To achieve this, multiple serial simulations were run simultaneously in the serial test cases. The concurrency of the serial simulations was equivalent to the concurrency of the parallel iterations for a pair of test cases (labelled S and P, respectively).

To assess how performance scales with additional concurrency and CPU cores, the tests were repeated with increasing concurrency until the number of concurrent simulations is twice the number of physical cores. In addition, the tests were repeated with Hyperthreading and Turbo-Boost disabled, to isolate the impacts of these technologies vs. increased CPU cores.

Measurement of test metrics was done through logging statements in the Tyche code. Time values were extracted from the log files at the end of each run.

2.3.3 Exploratory Testing Environment

Exploratory testing was performed on a laptop with the specifications listed in Table 1: Test Laptop - System Specifications.

[1] Note: I/O was not considered a significant factor because the parallel execution was specifically designed to run in an HPC environment using individual files for individual iterations to support error recovery functions.

Table 1: Test Laptop - System Specifications

Parameter	Value
Operating System	Windows® 10 Professional
CPUs	Intel CORE i7-7700HQ CPU @ 2.8Ghz
Sockets	1
Cores	4
Logical Processors	8
Cache	L1: 256 Kilobytes (KB) L2: 1.0 Megabytes (MB) L3: 6.0 MB
	16 Gigabytes (GB)

The following tools were used to do exploratory performance profiling:

Table 2: Tools Used for Exploratory Testing

Tool	Comments
Visual Studio (VS) Professional 2015	Visual Studio 2010 performance tools did not work on Windows® 10 because of changes to windows security model Some plugins required VS 2013 or VS 2015 (see below)
Windows Performance Monitor	Included with Windows®.
Windows Resource Monitor	Included with Windows®
Process Explorer	https://docs.microsoft.com/en-us/sysinternals/downloads/process-explorer
Concurrency Visualizer	Plug-in to Visual Studio Requires Visual Studio 2015 https://docs.microsoft.com/ en-
Visual Studio Performance Profiler	Built in tool with Visual Studio 2015 Pro
Microsoft Child Process Debugging Tool	Requires Visual Studio 2013 https://marketplace.visualstudio.com/items?itemName=Gregg Miskelly.MicrosoftChildProcessDebuggingPowerTool

2.3.4 Structured Test Configuration

Common test parameters were used for all tests as documented in Table 3.

Table 3: Simulation Test Parameters

Parameter	Value
Tyche Input file	Target Exploratory – Sept 16old.tyi
Force Structure(s)	2008
Years	7
Random Seed	-2147
Scenario Types	Scheduled, Random
Apply Specialized "Lift"	True

Testing was performed on a dual CPU 16 core windows server.
Specifications for the system are described in Table 4.

Table 4: Test Server - System Specifications

Parameter	Value
Operating System	Windows® 10 Professional
CPUs	Dual Intel XEON™ E5 2650 v2 2.6 Ghz
Sockets	2 (e.g. dual CPU)
Cores	16 (8 per CPU)
Logical Processors	32 (16 per CPU)
Cache	L1: 1 MB, L2: 4 MB, L3: 40 MB
Random Access Memory (RAM)	128 GB

2.3.5 Test Procedure

This section outlines the procedure followed for testing on the contractor test server. The exact steps to follow will differ on the HPC system.

To prepare the test machine, the following steps were required:

1. Install Tyche on the test machine.
2. Shutdown or disable any non-critical Windows services or applications that are consuming significant system resources.
3. Change Windows "Power and Sleep" settings to prevent screen from turning off, locking, or displaying slideshow while tests are running.
4. Change Windows "power plan" settings to ensure maximize performance. On a laptop, set to maximum performance, disable sleep, and ensure the laptop is plugged in.
5. Disable any anti-virus "on access" scan, or add Tyche processes as exclusions to any running anti-virus software.
6. Use a performance monitoring tool to ensure there is no unexpected load on system resources including CPU and memory.
7. Clear Tyche dashboard of all completed or aborted simulations. This will clear the Registry, which has a non-trivial impact on Tyche performance.

To execute the test cases presented in Section 2.3.6, the following steps were performed:

1. Open the Tyche Dashboard
2. Set "Maximum Concurrent Simulations" in the Tyche dashboard setting dialog to the

required Max Concurrent Simulations value for the given test.

3. For a parallel test, create and execute a single simulation using the parameters from Table 3: Simulation Test Parameters.
4. For a serial test, create multiple simulations using the parameters from Table 3: Simulation Test Parameters. The total number of running simulations should equal the required Max Concurrent Simulations value for the given test.
5. For serial tests, to synchronize starting of multiple serial simulations, first set Max Concurrent Simulations to 0, this will prevent any simulations from running. When all simulations have been launched and are waiting in the queue, change the value of Max Concurrent Simulations to the number required for the test.
6. Wait for Simulation(s) to complete. Do not use the test computer while the tests are running.
7. Extract performance logging data from each output log file Target Exploratory - Sept 16old.log.

2.3.6 Test Cases

The following test cases are proposed for running on the HPC system.

Table 5: Test Cases

Test Number	Serial or Parallel	Max Concurrent Simulation	Total Simulations Run	Iterations per Simulation	Total Iterations
1S	Serial	1	1	144	144
1P	Parallel	1	1	144	144
2S	Serial	2	2	144	288
2P	Parallel	2	1	144	144
3S	Serial	4	4	144	576
3P	Parallel	4	1	144	144
4S	Serial	8	8	144	1152
4P	Parallel	8	1	144	144
5S	Serial	16	16	144	2304
5P	Parallel	16	1	144	144
6S	Serial	24	24	144	3456
6P	Parallel	24	1	144	144
7S	Serial	32	32	144	4608
7P	Parallel	32	1	144	144
8S	Serial	48	48	144	6912
8P	Parallel	48	1	144	144

Notes:

1. Existing issues with the Tyche software prevented running more than 173 iterations with the given test inputs.
2. To control for the fact that individual iterations in simulations vary in computational complexity, it was important to run every test with the same number of iterations per simulation. This means that, in the serial case, many more total iterations were run. For example, 8 concurrent serial simulations (of 144 iterations each) were needed to load eight logical processors. The loading would be equivalent to 1152 (144 x 8) total iterations; versus one parallel simulation distributed as 144 fractional

simulations over 8 processes. Correspondingly the Simulation Run Times were longer for the serial cases, but the Average Iteration Times are directly comparable between serial and parallel cases.

2.3.7 Test Runs

The following test runs were performed on the contractor test server:

1. All Tests 1S/1P to 7S/7P
    a. Hyperthreading ON, Turbo-Boost ON
2. Parallel Tests 1P to 7P
    a. Hyperthreading OFF, Turbo-Boost OFF

When the serial tests were run, they would all conclude at nearly the same time and each simulation would attempt to complete input/output operations. Exploratory testing showed that the contention for disk resources subsequently affected the time measurements. Since the parallel testing did not exhibit this behaviour to the same extent, parallel testing was performed with Hyperthreading OFF and Turbo-Boost OFF to draw a comparison to the equivalent tests with Hyperthreading ON and Turbo-Boost ON. One could argue that a more thorough testing scheme would have tested Serial tests with Hyperthreading OFF and Turbo-Boost OFF, but this testing was not considered high value by the contractor and as such was omitted from the analysis.

The following test runs are proposed to be performed on the HPC system:

1. All Tests 1S/1P to 8S/8P

2.3.8 Results

The total **Simulation Run Time** measured for test runs was divided into 2 parts for performance measurement purposes:

1. **Iteration Time**: The elapsed time from the start of a simulation until the final iteration has completed. Iteration time includes all time spent in simulation setup and running iterations of the simulation, from the time Tyche SE processes are spawned until the final iteration completes.

2. **Post Processing Time**: The elapsed time starting when all iterations are completed and ending when the simulation run is completed. Post processing time includes merging output files, statistics generation and updating the risk spreadsheet.

The **Average Iteration Time** is Iteration Time multiplied by concurrency, divided by number of iterations. This is the consistent metric that can be used to compare performance between serial and parallel runs, and between varying parallel concurrency.

The results from test run 1 and run 2 on the test server are documented in Table 6 and Table **7** respectively.

Table 6: Test Results – Test Run 1

Test Number	Max. Concurrent	Simulation Run Time (s)	Iteration Time (s)	Post-Processing Time (s)	Iterations	Average Iteration Time(s)
1S	1	1164.4	1102.9	61.5	144	7.66
1P	1	1327.9	1266.1	61.8	144	8.79
2S	2	1170.5	1105.0	65.5	288	7.67
2P	2	688.4	626.9	61.4	144	8.71
3S	4	1265.3	1178.2	87.1	576	8.18
3P	4	392.7	331.4	61.4	144	9.21
4S	8	N/A	1383.5	N/A	1152	9.61
4P	8	251.7	188.8	62.9	144	10.5
5S	16	N/A	1493.8	N/A	2304	10.4
5P	16	166.6	104.8	61.8	144	11.6
6S	24	N/A	1989.8	N/A	3456	13.8
6P	24	151.6	90.3	61.3	144	15.1
7S	32	N/A	2538.0	N/A	4608	17.6
7P	32	161.5	86.2	75.4	144	19.2

NOTE: Simulation Run Time and Post-Processing Times were not measurable for some of the serial tests (marked as **N/A** in table) because synchronizing many serial tests caused contention in post processing phase that required manual intervention. This did not impact the Iteration Time measurement.

Annex C contains sample log files from which the data presented in Table 7.

Table 7: Test Results – Test Run 2[2]

Test Number	Max. Concurrent	Simulation Run Time (s)	Iteration Time (s)	Post-Processing Time (s)	Iterations	Average Iteration Time(s)
1P	**1**	1674.8	1599.3	75.5	144	11.1
2P	2	875.8	800.7	75.1	144	11.1
3P	4	489.9	412.3	77.6	144	11.5
4P	8	300.5	224.3	76.2	144	12.5
5P	16	199.3	122.3	77.0	144	13.6
6P	24	186.9	110.6	76.3	144	18.4
7P	32	195.8	109.3	86.5	144	24.3

Important note: Iteration Time results were adjusted from measured values in the parallel test cases to account for the fact that the last wave of iterations had a staggered completion time. This skewed the results especially in the case where the number of waves was small (e.g., 144/32=5 waves). To account for this, the average completion time of the last wave of iterations was taken as the final iteration completion time. For example, in Figure 1 below with 4 cores and 12 iterations, the simulation end time would be taken as 19.25s vs. 25s.

[2] For a concurrent serial test, all time values (Simulation Run Time, Iteration Time, Post Processing Time, Average Iteration Time) are taken as either an average or one representation sample of the *N* concurrent serial simulations. For Parallel Tests, there is only one simulation.

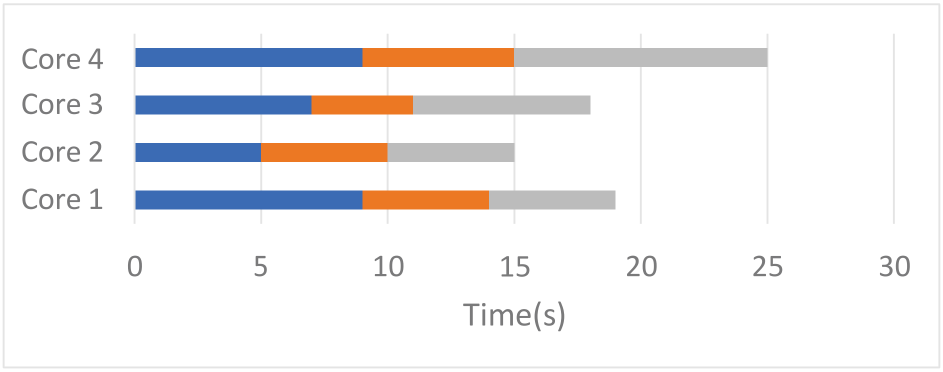

Figure 1: Staggered Iterations

2.3.8.1 *Future Considerations*

Ideally the number of iterations would be greatly increased so that the impact of the aforementioned variation in completion time would be reduced and thus could be ignored (e.g., run 1440 iterations vs. 144). However, this was not possible in the current code base, due to a known issue that prevented more than 174 iterations from running for the test input file under the specified input parameters.

An alternate testing scheme might simply have been to run more than one parallel simulation so that the cores were always fully loaded. This would have more closely resembled the serial case and provided more than one run to obtain average values.

2.3.9 Discussion of Results

The following points are discussed in the sections below:

- Serial versus parallel execution
- Degradation with concurrency
- Other performance results

2.3.9.1 *Serial versus Parallel Execution*

To compare performance of serial vs. parallel simulation runs (as defined in Section 2.3), *N* concurrent simulations (in the serial case – S tests) and *N* concurrent iterations (in the parallel case – P tests) were run in separate tests. The purpose was to load the same number of logical processors in each case. To compensate for differences in computational complexity of individual iterations, the same number of iterations were run for all simulations.

The data from Figure 2 and discussed subsequently shows that Average Iteration Time increases with concurrency for both serial and parallel simulations.

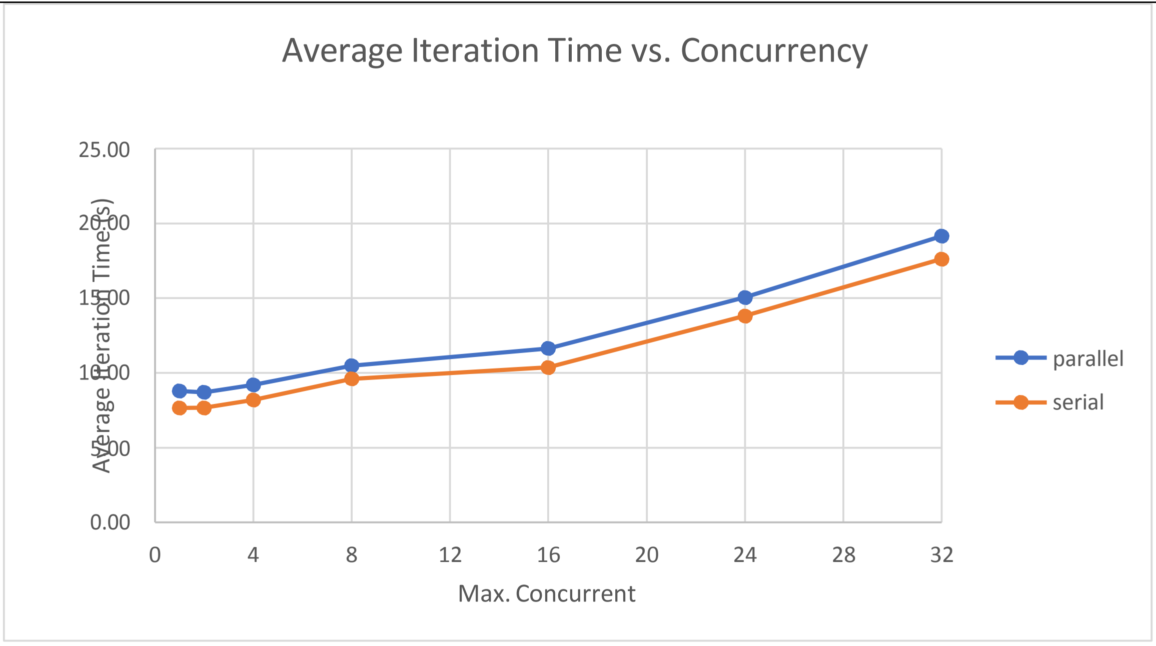

Figure 2: Average Iteration Time vs. Concurrency

The difference between the serial line and the parallel line in Figure 2 highlights the overhead of moving from serial to parallel execution. Exploratory testing revealed two categories of overhead:

1. **Process Coordination** – overhead associated with process coordination using the registry (e.g. Wait For Orders). This is significant only when Max. Concurrent is small (~1-4).
2. **Process Overhead** – overhead associated with executing each iteration in a separate process. This includes:

    a. Operating System creation and termination of the process and its threads.
    b. Loading of code into memory.
    c. Just in Time compilation (JIT) runs in each process, so the code is being JITed for each iteration
    d. Input files are parsed, output files are written
    e. Initialization and simulation setup code is run
    f. Data and variables are initialized

Process coordination is significant when the maximum number of concurrent simulations is relatively small, which is the case for most personal computing (PCs) devices. For the HPC version of Tyche, the HPC job manager will allow many jobs to be executed at the same time, so the overhead of process coordination on the HPC version of Tyche should not impact performance to the same extent. The process overhead is inherent in the design of jobs as separate processes, and will be similar on the HPC system.

Comparing the serial vs. parallel execution gives the results in Table 8.

Table 8: Parallel vs. Serial Overhead

Test Number	Max. Concurrent Loading (*N*)	Iteration Time (s) (*X*)	Performance Specification (*X/N*)*1.1	Parallel Specification Met	Increase in Iteration Time
1S	1	1102.9	1213.2		
1P	1	1266.1		FALSE	14.8%
2S	2	1105.0	607.8		
2P	2	626.9		FALSE	13.5%
3S	4	1178.2	324.0		
3P	4	331.4		FALSE	12.5%
4S	8	1383.5	190.2		
4P	8	188.8		TRUE	9.2%
5S	16	1493.8	102.7		
5P	16	104.8		FALSE	12.3%
6S	24	1989.8	91.2		
6P	24	90.3		TRUE	8.9%
7S	32	2538.0	87.2		
7P	32	86.2		TRUE	8.7%

As percentages, the increase in iteration time spans a range of 8.7% to 14.8% which implies a high degree of variance. No sensitivity analysis was completed, so it is unclear whether the variance is acceptable or not.

The test cases are independent and just over half fail, but not all fail, and in fact, the cases that are more representative of the HPC environment (4S to 7P), meet the performance requirement. Based on the testing conducted, it is fair to say that Tyche may meet specification under certain loading conditions. This will require further testing to determine once debugging has been completed and a full test with a large number of iterations can be run.

Based on the reduced degradation at higher maximum concurrent simulation settings (tests 4S to 7P), it is expected that the HPC parallel version will likely meet the requirement to run a parallel simulation on *N* logical processors in no more than a time of *X/N*1.1*, where X is the time taken for *N* equivalent concurrent serial simulations to run on the same *N* logical processors. As noted by the TA, a rigorous analysis of execution time should include statistical significance in the future.

For larger values of Max. Concurrency, as would be expected in an HPC system, the measured increase was below the 10% threshold. Case 1P is deemed unrealistic because parallel execution does carry some overhead, and so if only 1 maximum concurrent simulation is available, it would make more sense for the user to run the simulation in serial rather than parallel mode.

These percentages are very specific to the test data and input parameters. Most of the process overhead is relatively fixed and would not significantly change depending on the computational complexity of the iteration. For example, process overhead includes loading the configuration input file, loading and JITing the code, and writing the output files. None of these would increase or decrease considerably for a longer or shorter iteration. Therefore, the percentages calculated above would be greater for a shorter iteration and lesser for a longer iteration.

2.3.9.2 *Performance Degradation with Concurrency*

The preceding section discussed overhead of parallel execution vs. serial execution under the same CPU loading. In addition, it was observed that performance does not scale linearly with the number of logical processors (e.g. average iteration time does not remain constant). There is a degradation in both serial and parallel simulations to roughly the same extent. However, it appears that the degradation is not due to anything specific to Tyche code, but more general issues with multi-processing.

The suspected factors related to performance are discussed in this section, including:

1. Turbo-Boost
2. Hyperthreading
3. Symmetric Multiprocessor (SMP) architecture

2.3.9.2.1 Turbo-Boost [2]

On modern Intel CPUs (especially laptops) the CPU clock speed is dynamically adjusted based on the load on the CPU. This is done to reduce power consumption and heat when the CPU is not under load, and to overclock the CPU when under load. For example, on the test server, the base clock speed is 2.6 GHz, but when required it can overclock itself up to 130% (or 3.4 GHz). This capability is called Turbo-Boost (TB).

Comparing results from test run 1 and test run 2 show the impact of Turbo-Boost. For example, parallel test 1P with a concurrency of 1, had an average iteration time of 11s (TB disabled) vs. 8.79s (TB enabled). This corresponds to a speedup of 26% which is close to the maximum TB. This result also implies that Tyche scales linearly with an increase in clock speed.

It was observed during testing on the test server that clock frequency varied from a maximum of 126% when running with a concurrency of 1, down to 114% when concurrency was 32. Parallel execution of multiple concurrent simulations will appear slower because the TB will throttle itself from 126% down to 114%, due to temperature and load balancing between CPUs. Lower TB means degraded performance.
TB explains *approximately* 9% of the degradation in relative performance from 1 to 32, since the TB is reduced as the concurrency is increased.

2.3.9.2.2 Hyperthreading [3]

With Hyperthreading enabled, the operating system treats an 8 core hyperthreaded CPU as 16 logical processors. Each physical core is running 2 logical processors. The 2 logical processors per CPU Core share processing resources, but the processor state is physically duplicated (the CPU registers, etc.). This makes it possible for the CPU Core to quickly switch between 2 threads when it would otherwise stall. For example, on a cache miss, without hyperthreading the CPU Core would stall until the memory address could be retrieved from main memory. With hyperthreading, the CPU Core can switch to the other thread quickly because it already has the processor state. This is distinct from a traditional thread context switch that occurs when threads are time multiplexed on a single core non-hyperthreaded CPU.

The effect of hyperthreading can be seen in Table 6 specifically tests 5 and 7. As the maximum concurrent simulations exceeds the number of cores (16), the degradation in Average Iteration Time spikes. For example, serial test with concurrency 16 has an Average Iteration Time of 10.4s vs. 17.6s for serial test with concurrency of 32.

In test run 2, without hyperthreading, test 5P iteration time is 13.6s at 16 maximum concurrent simulations where test 7P is 24.3s at 32 maximum concurrent. The average iteration time nearly doubles (78.7% increase) between 16 and 32 as expected. .

2.3.9.2.3 SMP Architecture

Finally, even when Turbo-Boost (TB) and Hyperthreading (HT) are eliminated as factors, there remains additional degradation due to the nature of SMP computers (discussed in section 2.3.12 Multiprocessor Architecture). In Figure 3, until the total number of simulations run reaches the number of cores (16) there is a small degradation with concurrency.

For example, as can be seen from Table 7, at a concurrency of 1, the Average Iteration Time is 11.1s, at concurrency of 16 it is 13.6s. The throughput in iterations per second is:

- 1 core: 1/11.1 => 0.09 iterations per sec second
- 16 cores: 1/13.6*16 => 1.2 iterations per second

Therefore by employing 16 cores, the throughput is not simply 16 times greater than a single core, in this case the throughput is closer to **13 (1.2/0.09)**. Assuming the improvement in throughput would be 16 times greater than single core throughput, a throughput of 13 times would appear as an apparent degradation of 3 less than 16. This degradation is roughly in line with benchmarks. For comparison, the benchmarking site Geekbench [4] shows a multicore speedup of 6.36 for a single 8 core CPU of the type used on the test server. Since the test server has dual CPUs we can roughly double that to get **12.72** which is close to the **13** we measured with Tyche.

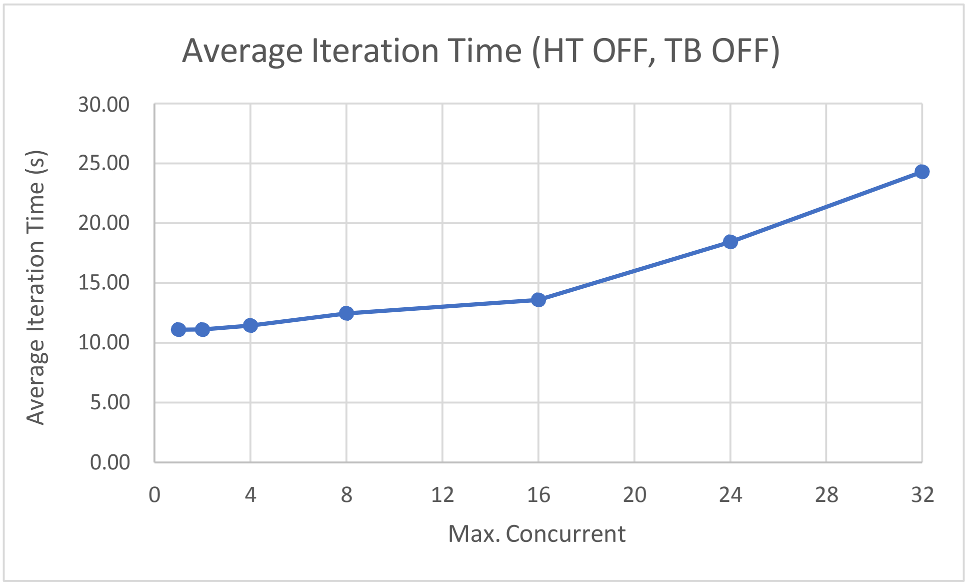

Figure 3: Average Iteration Time (HT/TB off)

2.3.10 Other performance results

As part of exploratory performance testing, some other performance areas were analyzed and ideas for improvement generated. While not part of the core task, they are documented here for reference. One change to Tyche implementation was made to improve the logging performance.

2.3.10.1 *Post Processing*

Results from tests run on a test laptop show the change in post-processing time with number of iterations.

Table 10: Post Processing Time Tests

Iterations	Post Processing Time(s)
2	44
10	45
25	46
50	48
100	51
150	55

Post processing time has a constant component independent of number of iterations, there is also a variable time that grows with number of iterations. In this test, it is about 7s per 100 iterations.

Profiling revealed that most of the Post Processing Time is spent processing and updating the Risk.xls spreadsheet. This is done through office interop, which uses Microsoft's component object model (COM) interop to spawn hidden instance(s) of excel. This requires inter-process communications through layers of COM objects and is known to perform poorly. In addition, the Tyche code iterates through Excel cells, updating each (vs. sending data in batches) which maximizes the inter-process communications.

On the test laptop, parallel execution file handling[3] was found to be an insignificant part of the post processing time. However, it is possible on the HPC system that when files are accessed over a network share, that the merging of files could be a bigger bottleneck, especially for a very large number of iterations.

[3] This is additional .tyi and .tyof file handling for fractional simulations.

2.3.10.2 *Inter-process Communication*

As referenced in the performance results above there is an overhead per iteration due to inter-process communication between the Tyche Simulation Manager (SM) and Tyche SE.

The coordination of processes is done through the registry. The SM periodically (on a 1 second timer) iterates through the registry and updates the orders. Similarly, waiting processes (also on 1 second timer) check the registry for orders.

One possible improvement that could be made would be to fire the timers more frequently. This will work to reduce the wait times, but it does have a side effect of increasing the processer overhead significantly for the registry polling code. A better solution is to change the inter-process communication mechanism to use synchronization objects [5] (e.g. semaphore) instead of the registry. This would eliminate the polling and lag times to zero; however, it would require extensive changes to the Tyche code.

2.3.10.3 *Logging*

Tyche logging implementation was inefficiently reading and rewriting the entire log file each time an entry was added to the log. The logging was modified to append to the end of the file.

2.3.10.4 *Native Image Generation*

Microsoft .NET has a tool called NGEN (Native Image Generation) [6] that will compile .NET byte code into native code for the machine thus bypassing the JIT compilation step for precompiled modules.   The benefit is that code can be shared between processes, and it vastly reduces the amount of JITing needed. The downside is that the natively compiled code must be maintained and there will be possible issues between different hardware/platforms. The original Visual C#.NET solution would remain available, but native images must be maintained.

This could possibly reduce the process start-up overhead since currently the JIT compiler and code loading is required for each spawned Tyche SE process. It may also give an overall code speedup in both the serial and parallel cases.

2.3.10.5 *Dashboard GUI*

Updating of progress bars on the Tyche SM dashboard was observed as very inefficient. For example, almost 20% of a logical processor was used to update 10 progress bars. The usage increases linearly with number of progress bars showing. Testing did not explore whether the performance changed in response to whether the SM was minimized or maximized.

2.3.11 Prior Testing Results

Results of testing [7] of a previous Tyche version from 2010 were made available to the contractor. The report indicated that concurrent processing should not have a significant effect on the average simulation time. Two observations from the report support that performance expectation:

- ".. is able to run one Tyche Simulation per core without significantly increasing total run time."
- "increasing the number of simulations equal to the number of cores does not seem to entail any performance decreases in the average run time"

The serial versus parallel testing done as part of this task showed that concurrent processing did have an effect on the average simulation time that was not insignificant. Taken at face value, if previous versions of Tyche were able to run without significantly increasing total run time, then it stands to reason that there were some changes between current and previous version environment and/or restructuring of the code base. This work did not explore previous versions of Tyche.

2.3.12 Symmetric Multiprocessor Architecture

As discussed previously, the testing carried out under this contract showed that Tyche processing did not, overall, scale linearly with number of cores. Average iteration time degrades with increasing concurrency. Benchmarks and exploratory testing did not clearly reveal causes inherent in the computer systems. However, previous tests showed that Tyche 2.3.4 did scale linearly and the architecture was not a factor. In the previous tests, HT and TB both existed, and multiple processors were used. In light of the current and previous tests, the common element between the two is a code base change to C#.NET and a restructuring of the code base to utilize native array-based variables, rather than objects and collections. It would seem reasonable to conclude that Tyche 3 in Visual C#.NET is susceptible to degradation with increasing concurrency.

> From Wikipedia: "**Symmetric multiprocessing** (**SMP**) involves a multiprocessor computer hardware and software architecture where two or more identical processors are connected to a single, shared main memory, have full access to all input and output devices, and are controlled by a single operating system instance that treats all processors equally, reserving none for special purposes. Most multiprocessor systems today use an SMP architecture. In the case of multi-core processors, the SMP architecture applies to the cores, treating them as separate processors." [8]

In the case of multi-core systems, such as the test server, there are multiple physical CPU COREs within the same chip (e.g. Replace "processor" with "CORE" in Figure 4). As with other SMP systems, the system resources such as Main Memory, system buses and I/O, are shared between the COREs. In addition, for a multi-CORE machine, the level 2 cache is also shared between cores (e.g., replace the multiple caches in SMP diagram with a single shared cache).

Since Tyche has minimal I/O (during iteration running), suspected degradation is most likely due to contention for memory (main memory and L2 cache).

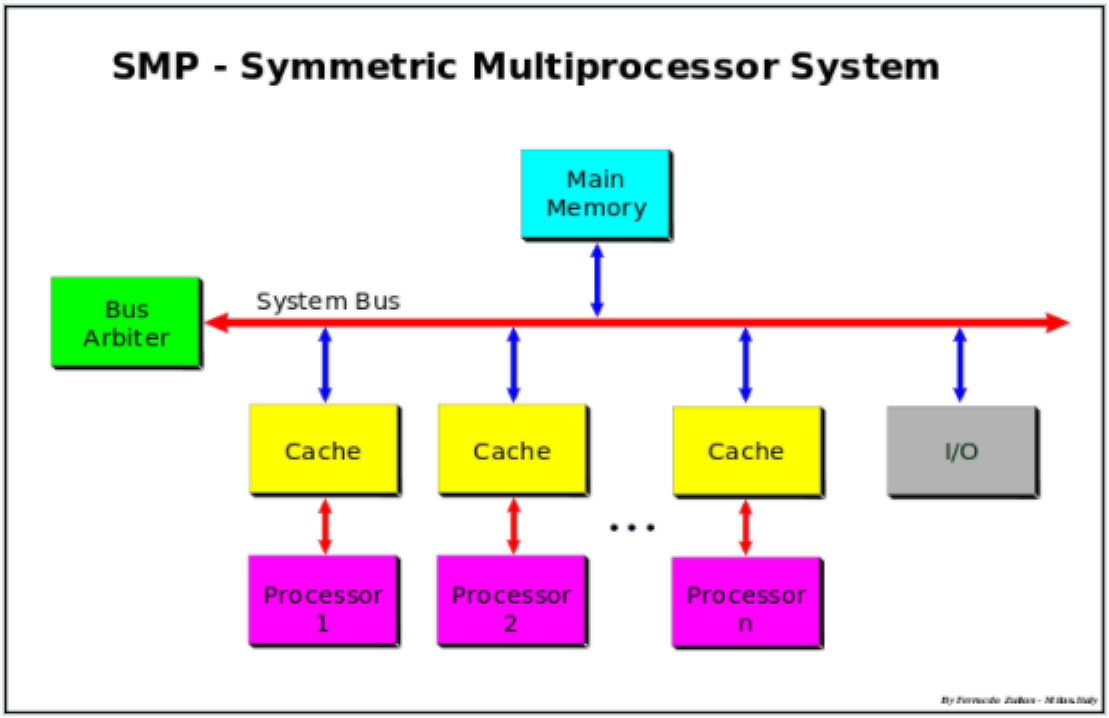

Diagram of a symmetric multiprocessing system

Figure 4: SMP Diagram [8]

2.3.13 Summary

A test plan was developed to test the parallel and serial functionality of Tyche. The test plan was run on the contractor system and the results compiled. The same tests are to be performed by the TA on the HPC system. Based on the results, the expectation is that the parallel version of Tyche will be, on average, within the 10% overhead of the serial version on the HPC system for the given test inputs.

The observed 10% overhead was related to overhead of running each iteration in its own process. There does not appear to be anything in the core simulation code which would limit the scalability of parallel simulations. Exploratory testing did reveal that the post-processing code will, however, increase processing time linearly with number of iterations and may become an issue at very high iterations. Since the tests were constrained to 144 iterations, the exact relationship between number of iterations and post-processing time was not measured. Unfortunately, an existing software issue limited the ability to fully test scalability to very large numbers of iterations which introduce some risk that undiscovered scalability issues may be latent in the software.

Multi-processor scalability of Tyche was determined to be roughly equivalent between the parallel and serial version. In addition, the expected scalability based on the characteristics of multicore hyperthreaded CPUs was estimated and can be used to extrapolate the expected scalability on the HPC system.

Finally, some other areas of potential performance improvement were documented for future reference.

2.4 Task 3 & 4 – Update Compiled Help (CHM) and User Guide

The compiled help and User Guide are generated from a single source document that is authored in LaTeX (tex). The LaTeX document is run through a publishing process that results in two end media formats: a User Guide in portable document format (PDF), and a compiled help in CHM. The content in both the CHM and PDF is identical but the formatting of the content is slightly different to accommodate the end media formats.

Under this contract, an edition numbering scheme was devised and added to the introduction section of the document. The introduction section is the earliest section that is common to both end media formats.

The documentation from Tyche 3.1 was marked up and provided to the contractor to update the documentation for Tyche 3.2. All the edits except for one were completed. The outstanding edit asked for the LaTex tables to be re-inserted into the source document. During the documentation process for Tyche 3.1, it was found that the LaTeX description of tables rendered well in PDF, but not well in CHM. The work-around was to create the tables in Microsoft Excel®, print to an image, and insert the images into the LaTeX document. Images inserted in LaTeX render well in in both the PDF and CHM.

It is possible that with some trial and error using different LaTeX packages that a suitable table definition could be found that will render LaTeX tables nicely in both PDF and CHM. This work could be done, but it is a matter of preference. There will always be some content (such as Visio diagrams) that will live in external documents that will have to be rendered to an image before being used in the PDF and CHM.

For those comfortable working in LaTeX, the source Tyche.tex file was made available for markup. A “trackchanges” package is commented out in the LaTeX document but can be re-enabled to use the LaTeX track changes feature. For those that are not comfortable working in LaTeX, the PDF has been converted to DOCX format so that Microsoft Word® track changes can be used. Note that the conversion process is only *nearly* perfect, so the PDF version has to be read side by side with the DOCX to ensure conversion errors are not flagged as documentation errors.

Additional images were added to the documentation. Whereas the documentation used to have “refer to red dot number 1 on image…”, to call the reader’s attention to a specific area of a GUI, instead, a clipped copy of the highlighted area was inserted into the document.

The word “days” (or similar variants) was still found in the document. Those instances have been replaced with the term “mark” to reflect the adjustable time scale.

2.5 Task 5 – Contractor Report

No additional commentary.

3. OVERALL PROJECT ISSUES

3.1 Overview

This section provides an overview of issues that arose during the project, and how they were mitigated.

3.2 Issue 1: SelAssets.cs Not Runnable in Version 3.1+

Following the delivery of Tyche 3.1, a couple of observations were found in SelAssets.cs:

1. Some of the timescale conversions were missing in the original 3.1 delivery.
2. Some of the base lookup logic was not performing as expected.

To maintain source code control, the received 3.1 code was marked internally as version 3.1.1.

Tyche version 3.1.1 would not allow Task 2 – Propose and Implement Performance Test to progress and caused issues with Task 1 – Fix Bugs.

Mitigation: Under the client's guidance, the contractor replaced the Tyche 3.1.1 SelAssets.cs with the Tyche 3.1 version of SelAssets.cs. Minor bug fixes occurred until a runnable version of Tyche permitted Task 1 – Fix Bugs and Task 2 – Propose and Implement Performance Test to progress. The runnable version of Tyche was released as Tyche 3.1.2 to the client on Dec 14, 2017 as an un-scheduled release. The SelAssets.cs issue had a low overall impact and was quickly resolved.

3.3 Issue 2: Visual Studio 2010 Performance Profiling Not Workable

The performance profiling features available in Visual Studio 2010 were incompatible with Windows 10 due to changes to the windows security model. Microsoft documentation indicates that Visual Studio 2010 profiling is compatible with older versions of Windows.

Mitigation: Installed Visual Studio 2015 on the Windows 10 computer rather than install an older operating system; determined that Visual Studio 2015 was compatible with the client system.

4. RECOMMENDATIONS FOR FUTURE WORK

4.1 Tyche development support

To lower the cost of future upgrades and maintenance, a maintenance upgrade consisting of a code re-factoring to include unit and integration tests (SpecFlow [9], Gherkin [10] and executable specifications [11] are highly recommended). This will mitigate design fatigue as new features are added.

Recommended tests would focus on specific application behaviour that is high valuable and easily automated. The focus would be on algorithmic items such as bumping, travel time, application of lift, etc. and not on User Interface (UI) items (UI tests are known to be brittle and expensive to maintain). The goal would be to have a suite of tests that could be run following any bug fix that might affect key algorithms.

The key to robust executable specifications is to test behaviours and to resist the urge to test specific implementations. SpecFlow (through Cucumber [12]) offers a unique approach for loosely coupling behaviour tests to the existing code base using feature files. For example, if feature files existed for Tyche 2.3.4, they could have been used, without any changes, in later implementations (such as Tyche 3.0 and later) even though the underlying implementation changed significantly. This highlights the utility of executable specifications. They become a “source of truth” as different implementations achieve the same behaviour.

Feature files contain the specification, but to make the specification executable, the feature files are accompanied by Step Definition Files [10] (to keep this section concise, it is assumed the reader will use the linked information to get the background on these concepts). The step definition files are coupled to the code, and as a result, maintaining the step definitions is the true cost to the project of adding executable specifications.

The ideal time to introduce executable specifications in a software project is before the first line of code is written, however this rarely happens in practice. The next best time to introduce executable specifications is when design fatigue begins to appear in a software project. Design fatigue is the point at which even small changes take a long time, and/or are highly error prone due to the programmer not knowing the full consequence of each change, at the time of making the code. Executable specifications allow a programmer to make a change, re-run the executable specifications, and determine immediately if the change has un-intentionally and adversely affected any other behaviours.

Executable specifications are tests, and test writing requires effort. The level of effort attributed to writing tests is directly proportional to the criticality of the failure when the user is using the software, or to the size of the loss of the historical investment when a feature is broken in a new release. Tyche would fall into the latter category. Tyche has been around for years and has tried-and-true features such as bumping, travel time calculations etc. that need to work release after release. To ensure the tried-and-true features remain tried-and-true, each feature would be covered by one or more executable specifications that would alert a programmer immediately if an earlier feature has been broken. Writing executable specifications is largely a time and material activity.

There is no hard and fast rule about how much time to invest into executable specifications, it is largely a subjective assessment. In the case of Tyche, the contractor recommends a first round investment equal to a typical “ramp-up” time for a new developer. A new developer should technically read the entire user guide and read through all the code before making any changes. Given the complexity of Tyche, this is likely a 40 to 60 hour effort. A first round investment equal to 40 to 60 hours is likely a good first investment into executable specifications; with up to two more rounds likely required to achieve an acceptable level of test coverage.

4.2 Functional changes to Tyche

Functional changes for Tyche could include the following:

1. Full integration of the risk analysis, so that no user intervention is required to set up the Excel Risk file before the simulation run.
2. Produce an optimization wrapper in Microsoft Visual C#.NET (compatible with Microsoft Visual Studio 2010 Premium) for the most up-to-date version of Tyche, such that all existing functionality is maintained while:
    a. The content for the political risk Excel spreadsheet [AD4] is generated automatically from the input/output data (based on version of Microsoft Excel®). The total political risk becomes one of the objectives for the optimization.
    b. The forces structure size (number of assets) from the selected force structure for analysis is calculated as a second objective for optimization.
    c. The option to add additional objectives must be built into the optimizer, through the user of additional spreadsheet input (e.g. cost per asset to produce a total cost for the force structure).
    d. The optimizer must use the composition of the force structure (number of assets of each asset type and location for basing of each asset) as the variables for the optimization. The scheduling offset can either be calculated using a function or included as a variable for the optimization. The function for choosing the scheduling offset must not preclude future input from a set of business rules to optimize asset distribution between locations.
    e. The optimizer must utilize one algorithm for multi-objective optimization appropriate to the problem at hand. The algorithm will be selected by the TA in collaboration with the contractor, based upon detailed examination of the problem type and simulation performance. The option to include additional algorithms in the future should be included in the design of the optimizer; and
3. As matter of normal development, the user guide and help files will require continuous updating as the application continues to evolve.

ANNEX A. TYCHE 3.2 FAMILIARIZATION

The Tyche documentation (Tyche.pdf) was updated based on the work undertaken through the contract. For a user familiar with Tyche, the following portions of the user guide would assist in becoming familiar with the updates resulting from the Tyche 3.2 work:

- Abstract
- Acknowledgements
- Introduction
- 1.1 Development
- 1.1.4 Requirements for New Development
- 1.1.5 Conclusions and Future Work
- 1.3.1 Capabilities of Tyche
- 2.2.1 Tyche Installation
- 3.1 About Window
- 3.2 Parent Window
- 3.3.1.1 Simulation TimeScale
- 3.3.2 Capabilities
- 3.3.5.3 Asset Levels
- 3.3.5.3.3 Level Type
- 3.3.5.3.4 Duration
- 3.3.5.4 Multi-Level Constraints
- 3.3.5.5 Asset Type Bump Table
- 3.4 Run Environment
- 3.4.1 Run in Debug Mode
- 3.4.2 Run Simulation
- 3.4.2.5 Execution Type
- 3.4.2.9 Run Button
- 3.4.2.10 Pause Button
- 3.4.2.11 Stop Button
- 3.4.2.12 Create HPC Job
- 3.4.2.13 HPC Job Form
- 3.4.2.13.1 Tyche Input File to 3.4.2.13.4 Generate HPC
- 3.4.3.3 Assigning and registering Assets to events
- 4.1 The Tyche Simulation Editor
- 4.1.2 Output Directory
- 4.1.7 Execution Type
- 4.1.10 Simulation Control Buttons
- 4.2.2 Dashboard window management
- 4.2.7 Troubleshooting and Logging
- [38] in Bibliography
- B.1 TYI Files (.tyi)
- Table E.1: Log Entries index
- G.2.2 Logging Class
- (all of) G.2.4 Sequence Diagrams
- G.4.2.3 Testing
- G.4.3 HPC
- G.5.1 Documentation Toolchain
- G.5.3 Major Documentation Changes

ANNEX B. PROJECT DETAILS

B.1 STATEMENT OF WORK (SOW) TASKS

6.1 The contractor must perform minor bug fixes and/or code/GUI improvements as identified by the TA or the contractor, and mutually agreed upon, during the course of the task. This includes, but is not limited to:

a. Ensuring abort and timeout functionality of serial and parallel simulations remain consistent with Tyche version 3.0 via the Simulation Manager, the Windows Task Manager, and the Windows Registry.
b. Ensuring the parallel simulation XML file generation is consistent, especially when errors occur, with the serial output case.
c. Bug fixes to the GUI upon Save File, Open Ideal Assets, XML Edit, scrollbar issues with the OpSched Viewer, and other unhandled exceptions.

6.2 The contractor must propose and implement a test plan for determining the functionality of the HPC implementation of Tyche in both parallel and serial mode. Any bug fixes and code changes that are required to meet the following performance specifications must also be carried out:

a. The test plan on the HPC system (to be mutually agreed upon) will be carried out by the TA, and must include the minimum number of steps to test out the parallel and serial functionality on the HPC system. The parallel version must produce the same output as the serial version (identical .tyo (zipped), .tya., .tyc, .tys., and .xls files) with no compiled or run time errors for the test input (.tyi) file(s) provided by the TA.
b. At full CPU loading on *N* CPUS, *N* serial simulations should complete in *X* time (when measuring the total simulation run time, from the time the simulation log files record the simulation as starting to the time all iterations are recorded as completed). A single simulation of the parallel version at full CPU loading on *N* CPUs must not take longer to run than (*X/N)*1.1* (or no more than a 10% increase in total run time than the serial version). A sample input file (.tyi) will be provided by the TA upon task start-up to test the performance of the system before and after on the contractor's system.

6.3 The contractor must update the compiled hyper text markup language (HTML) help file (.chm) whose sub-topics must be linked to the context-sensitive help in the Tyche Version 3.2 Visual C#.NET code. The help file must utilize the table of contents structure and topic content in [AD3] and [AD1]; with modifications permitted under Technical Authority (TA) approval where code functionality changes between versions. The appropriate locations in the Visual C#.NET code must be updated with new file names for sub-topics (HTML files) when the documents are completed. A list of changes to the code shall be maintained and delivered to permit tracking of effort and version control.

6.4 The contractor must produce an updated version of [AD3], meaning that all text and figures must reflect the most recent version of the Tyche code [AD1] in terms of graphical user interface, functionality, simulation engine operation, input/output data and any other factors determined in agreement between the TA and the contractor. The table of contents structure and topic content must reflect the help file content, such that the two documents are intrinsically generated from the same text and images. Any changes made at the final stages of editing of the written document must also be reflected in the help files.

6.5 The contractor must document the results of their work in a report.
All government furnished information and equipment will be made available on the first day of the contract.

ANNEX C. SOURCE DATA – TEST 2

The source data is provided via a zip file that contains one folder for each test run within test 2. Source data for test 6P (Parallel 24) and 7P (Parallel 32) were omitted for reasons of brevity. The contents of the zip are shown in Figure 5: Source Data for Test 2.

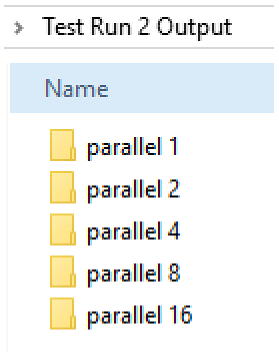

Figure 5: Source Data for Test 2

The number of each folder follows the maximum concurrent simulations setting for tests 1P to 5P.

Each folder contains a single log file (shown in Figure 6: Log File with Source Data) that contains the source data.

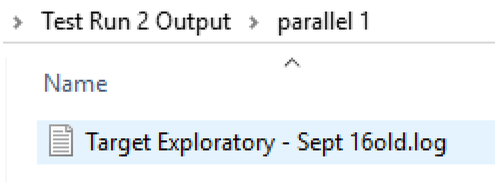

Figure 6: Log File with Source Data

The zip contains the complete source data. The following sections contain relevant extracts of each log file from each folder.

Note: To maintain brevity, only the first and last iteration log messages are provided.

C.1 PARALLEL 1 – TEST 1P

```
<log date="2018-01-24" time="19-26-29.804">RunTycheSimulationFromEditor - XML File written</log>
<log date="2018-01-24" time="19-26-29.885">RunTycheSimulationFromEditor - Parallel iteration files written</log>
<log date="2018-01-24" time="19-26-29.963">RunTycheSimulationFromEditor - Parallel processes launched</log>
<log date="2018-01-24" time="19-53-07.589">starting tyof merge</log>
<log date="2018-01-24" time="19-53-08.182">tyof merge complete</log>
<log date="2018-01-24" time="19-53-08.198">
        <!-- parallel log files -->
                <!-- Iteration_1.log -->
        <log date="2018-01-24" time="19-26-30.151">Simulation Starting: My unique ID is Node GRBD 233</log>
        <log date="2018-01-24" time="19-26-39.899">Iteration execution time (ms) : <exectime>9732.6152</exectime></log>
        <log date="2018-01-24" time="19-26-39.899">Simulation Completed</log>

… iterations 2 through 143 removed for brevity

                <!-- Iteration_144.log -->
        <log date="2018-01-24" time="19-35-20.314">Simulation Starting: My unique ID is Node GRBD 233</log>
        <log date="2018-01-24" time="19-35-31.736">Iteration execution time (ms) : <exectime>11406.5354</exectime></log>
        <log date="2018-01-24" time="19-35-31.736">Simulation Completed</log>
        </log>
<log date="2018-01-24" time="19-53-08.198">All iterations complete. (ms) : 1598219.3082</log>
<log date="2018-01-24" time="19-53-09.229">Statistics Generation Starting</log>
<log date="2018-01-24" time="19-53-09.245">Built the assets used for collection and calculation</log>
<log date="2018-01-24" time="19-53-09.261">Built the phases used in collection and calculation</log>
<log date="2018-01-24" time="19-53-40.246">Built the data structure used in collection and calculation of risk</log>
<log date="2018-01-24" time="19-53-42.636">Collected data from output file</log>
<log date="2018-01-24" time="19-53-42.636">Completed Asset Statistics Generation</log>
<log date="2018-01-24" time="19-53-42.652">Completed Scenario Statistics Generation</log>
<log date="2018-01-24" time="19-53-42.652">Completed Capability Statistics Generation</log>
<log date="2018-01-24" time="19-54-15.532">Completed Risk Spreadsheet Update</log>
<log date="2018-01-24" time="19-54-15.532">Statistics Generation Completed</log>
<log date="2018-01-24" time="19-54-24.735">Output file was successfully zipped</log>
```

C.2 PARALLEL 2 – TEST 2P

```
<log date="2018-01-24" time="19-04-10.420">RunTycheSimulationFromEditor - XML File written</log>
<log date="2018-01-24" time="19-04-10.639">RunTycheSimulationFromEditor - Parallel iteration files written</log>
<log date="2018-01-24" time="19-04-10.826">RunTycheSimulationFromEditor - Parallel processes launched</log>
<log date="2018-01-24" time="19-17-31.977">starting tyof merge</log>
<log date="2018-01-24" time="19-17-32.540">tyof merge complete</log>
<log date="2018-01-24" time="19-17-32.555">
        <!-- parallel log files -->
                <!-- Iteration_1.log -->
        <log date="2018-01-24" time="19-04-10.904">Simulation Starting: My unique ID is Node VDAA 543</log>
        <log date="2018-01-24" time="19-04-20.592">Iteration execution time (ms) : <exectime>9672.118</exectime></log>
        <log date="2018-01-24" time="19-04-20.592">Simulation Completed</log>

… iterations 2 through 143 removed for brevity

                <!-- Iteration_144.log -->
        <log date="2018-01-24" time="19-08-36.203">Simulation Starting: My unique ID is Node VDAA 543</log>
        <log date="2018-01-24" time="19-08-47.609">Iteration execution time (ms) : <exectime>11390.9189</exectime></log>
        <log date="2018-01-24" time="19-08-47.609">Simulation Completed</log>
        </log>
<log date="2018-01-24" time="19-17-32.555">All iterations complete. (ms) : 801729.2397</log>
<log date="2018-01-24" time="19-17-33.227">Statistics Generation Starting</log>
<log date="2018-01-24" time="19-17-33.243">Built the assets used for collection and calculation</log>
<log date="2018-01-24" time="19-17-33.259">Built the phases used in collection and calculation</log>
<log date="2018-01-24" time="19-18-04.041">Built the data structure used in collection and calculation of risk</log>
<log date="2018-01-24" time="19-18-06.431">Collected data from output file</log>
<log date="2018-01-24" time="19-18-06.431">Completed Asset Statistics Generation</log>
<log date="2018-01-24" time="19-18-06.431">Completed Scenario Statistics Generation</log>
<log date="2018-01-24" time="19-18-06.447">Completed Capability Statistics Generation</log>
<log date="2018-01-24" time="19-18-39.122">Completed Risk Spreadsheet Update</log>
<log date="2018-01-24" time="19-18-39.122">Statistics Generation Completed</log>
<log date="2018-01-24" time="19-18-48.289">Output file was successfully zipped</log>
```

C.3 PARALLEL 4 – TEST 3P

```
<log date="2018-01-24" time="18-52-36.998">RunTycheSimulationFromEditor - XML File written</log>
<log date="2018-01-24" time="18-52-37.077">RunTycheSimulationFromEditor - Parallel iteration files written</log>
<log date="2018-01-24" time="18-52-37.264">RunTycheSimulationFromEditor - Parallel processes launched</log>
<log date="2018-01-24" time="18-59-34.004">starting tyof merge</log>
<log date="2018-01-24" time="18-59-34.567">tyof merge complete</log>
<log date="2018-01-24" time="18-59-34.598">
        <!-- parallel log files -->
                <!-- Iteration_1.log -->
        <log date="2018-01-24" time="18-52-37.498">Simulation Starting: My unique ID is Node EMBF 744</log>
        <log date="2018-01-24" time="18-52-48.389">Iteration execution time (ms) : <exectime>10859.6453</exectime></log>
        <log date="2018-01-24" time="18-52-48.389">Simulation Completed</log>

… iterations 2 through 143 removed for brevity

                <!-- Iteration_144.log -->
        <log date="2018-01-24" time="18-54-48.674">Simulation Starting: My unique ID is Node EMBF 744</log>
        <log date="2018-01-24" time="18-55-00.346">Iteration execution time (ms) : <exectime>11640.9209</exectime></log>
        <log date="2018-01-24" time="18-55-00.346">Simulation Completed</log>
        </log>
<log date="2018-01-24" time="18-59-34.598">All iterations complete. (ms) : 417318.309</log>
<log date="2018-01-24" time="18-59-34.864">Statistics Generation Starting</log>
<log date="2018-01-24" time="18-59-34.879">Built the assets used for collection and calculation</log>
<log date="2018-01-24" time="18-59-34.895">Built the phases used in collection and calculation</log>
<log date="2018-01-24" time="19-00-06.318">Built the data structure used in collection and calculation of risk</log>
<log date="2018-01-24" time="19-00-08.771">Collected data from output file</log>
<log date="2018-01-24" time="19-00-08.786">Completed Asset Statistics Generation</log>
<log date="2018-01-24" time="19-00-08.786">Completed Scenario Statistics Generation</log>
<log date="2018-01-24" time="19-00-08.802">Completed Capability Statistics Generation</log>
<log date="2018-01-24" time="19-00-42.630">Completed Risk Spreadsheet Update</log>
<log date="2018-01-24" time="19-00-42.630">Statistics Generation Completed</log>
<log date="2018-01-24" time="19-00-52.443">Output file was successfully zipped</log>
```

C.4 PARALLEL 8 – TEST 4P

```
<log date="2018-01-24" time="18-45-44.822">RunTycheSimulationFromEditor - XML File written</log>
<log date="2018-01-24" time="18-45-44.916">RunTycheSimulationFromEditor - Parallel iteration files written</log>
<log date="2018-01-24" time="18-45-45.306">RunTycheSimulationFromEditor - Parallel processes launched</log>
<log date="2018-01-24" time="18-49-32.812">starting tyof merge</log>
<log date="2018-01-24" time="18-49-33.359">tyof merge complete</log>
<log date="2018-01-24" time="18-49-33.390">
        <!-- parallel log files -->
                <!-- Iteration_1.log -->
        <log date="2018-01-24" time="18-45-46.416">Simulation Starting: My unique ID is Node XXHF 736</log>
        <log date="2018-01-24" time="18-45-57.385">Iteration execution time (ms) : <exectime>10953.3976</exectime></log>
        <log date="2018-01-24" time="18-45-57.385">Simulation Completed</log>

… iterations 2 through 143 removed for brevity

                <!-- Iteration_144.log -->
        <log date="2018-01-24" time="18-46-55.793">Simulation Starting: My unique ID is Node XXHF 736</log>
        <log date="2018-01-24" time="18-47-08.324">Iteration execution time (ms) : <exectime>12515.9406</exectime></log>
        <log date="2018-01-24" time="18-47-08.324">Simulation Completed</log>
        </log>
<log date="2018-01-24" time="18-49-33.390">All iterations complete. (ms) : 228083.8876</log>
<log date="2018-01-24" time="18-49-34.015">Statistics Generation Starting</log>
<log date="2018-01-24" time="18-49-34.031">Built the assets used for collection and calculation</log>
<log date="2018-01-24" time="18-49-34.047">Built the phases used in collection and calculation</log>
<log date="2018-01-24" time="18-50-05.360">Built the data structure used in collection and calculation of risk</log>
<log date="2018-01-24" time="18-50-07.735">Collected data from output file</log>
<log date="2018-01-24" time="18-50-07.735">Completed Asset Statistics Generation</log>
<log date="2018-01-24" time="18-50-07.751">Completed Scenario Statistics Generation</log>
<log date="2018-01-24" time="18-50-07.751">Completed Capability Statistics Generation</log>
<log date="2018-01-24" time="18-50-41.006">Completed Risk Spreadsheet Update</log>
<log date="2018-01-24" time="18-50-41.006">Statistics Generation Completed</log>
<log date="2018-01-24" time="18-50-50.225">Output file was successfully zipped</log>
```

C.5 PARALLEL 16 – TEST 5P

```
<log date="2018-01-24" time="18-32-16.084">RunTycheSimulationFromEditor - XML File written</log>
<log date="2018-01-24" time="18-32-16.162">RunTycheSimulationFromEditor - Parallel iteration files written</log>
<log date="2018-01-24" time="18-32-16.912">RunTycheSimulationFromEditor - Parallel processes launched</log>
<log date="2018-01-24" time="18-34-23.900">starting tyof merge</log>
<log date="2018-01-24" time="18-34-24.462">tyof merge complete</log>
<log date="2018-01-24" time="18-34-24.478">
        <!-- parallel log files -->
                <!-- Iteration_1.log -->
        <log date="2018-01-24" time="18-32-17.584">Simulation Starting: My unique ID is Node LQSG 748</log>
        <log date="2018-01-24" time="18-32-28.897">Iteration execution time (ms) : <exectime>11297.1599</exectime></log>
        <log date="2018-01-24" time="18-32-28.897">Simulation Completed</log>

… iterations 2 through 143 removed for brevity

                <!-- Iteration_144.log -->
        <log date="2018-01-24" time="18-32-53.460">Simulation Starting: My unique ID is Node LQSG 748</log>
        <log date="2018-01-24" time="18-33-07.195">Iteration execution time (ms) : <exectime>13703.4687</exectime></log>
        <log date="2018-01-24" time="18-33-07.195">Simulation Completed</log>
        </log>
<log date="2018-01-24" time="18-34-24.478">All iterations complete. (ms) : 127550.0948</log>
<log date="2018-01-24" time="18-34-24.978">Statistics Generation Starting</log>
<log date="2018-01-24" time="18-34-24.993">Built the assets used for collection and calculation</log>
<log date="2018-01-24" time="18-34-25.009">Built the phases used in collection and calculation</log>
<log date="2018-01-24" time="18-34-56.415">Built the data structure used in collection and calculation of risk</log>
<log date="2018-01-24" time="18-34-58.821">Collected data from output file</log>
<log date="2018-01-24" time="18-34-58.821">Completed Asset Statistics Generation</log>
<log date="2018-01-24" time="18-34-58.837">Completed Scenario Statistics Generation</log>
<log date="2018-01-24" time="18-34-58.837">Completed Capability Statistics Generation</log>
<log date="2018-01-24" time="18-35-32.783">Completed Risk Spreadsheet Update</log>
<log date="2018-01-24" time="18-35-32.783">Statistics Generation Completed</log>
<log date="2018-01-24" time="18-35-41.986">Output file was successfully zipped</log>
```

REFERENCES

[1] SimFront Simulation Systems Corp. (Ed.) (2017), "Tyche 3.1 Help Files (electronic edition)," Defence Research & Development Canada Centre for Operational Research and Analysis. Updated from Avery, L., (2013), Tyche 3.0 help files (electronic edition), Eisler, C., Allen, D., Forget, A., Heppenstall, D., and Michalowski, P., (2009), Tyche 2.3 help.

[2] Intel, "Intel Turbo Boost Technology 2.0," Intel, [Online]. Available: https://www.intel.com/content/www/us/en/architecture-and-technology/turbo-boost/turbo-boost-technology.html#. [Accessed 16 03 2018].

[3] Wikipedia, "Hyper-Threading," 22 January 2018. [Online]. Available: https://en.wikipedia.org/wiki/Hyper-threading. [Accessed 16 March 2018].

[4] Geekbench, "Intel Xeon E5-2650 v2 Benchmarks," [Online]. Available: Intel Xeon E5-2650 v2 Benchmarks. [Accessed 16 March 2018].

[5] Microsoft, "Synchronization Objects," [Online]. Available: https://msdn.microsoft.com/en-us/library/windows/desktop/ms686364(v=vs.85).aspx. [Accessed 16 March 2018].

[6] Microsoft, "Ngen.exe (Native Image Generator)," [Online]. Available: https://docs.microsoft.com/en-us/dotnet/framework/tools/ngen-exe-native-image-generator. [Accessed 16 March 2018].

[7] "Benchmark Testing Using Tyche. From Personal Computer to Desktop Supercomputers.," Defence Research & Development Canada Centre for Operational Research and Analysis. Cheryl Eisler. Report DRDC CORA TN 2010-251, (2010).

[8] Wikipedia, "Symmetric multiprocessing", [Online]. Available: https://en.wikipedia.org/wiki/Symmetric_multiprocessing [Accessed 23 April 2018].

[9] specflow, "Home Page," [Online]. Available: http://specflow.org/. [Accessed 16 March 2018].

[10] Cucumber Ltd, "Reference," Cucumber Ltd, [Online]. Available: https://cucumber.io/docs/reference. [Accessed 16 March 2018].

[11] Pluralsight, "Executable Specifications: End-to-End Acceptance Testing With Spec Flow," [Online]. Available: https://www.pluralsight.com/courses/executable-specifications-specflow. [Accessed 16 March 2018].

[12] Cucumber Ltd, "Home Page," [Online]. Available: https://cucumber.io/. [Accessed 16 March 2018].

DOCUMENT CONTROL DATA

*Security markings for the title, authors, abstract and keywords must be entered when the document is sensitive

1. ORIGINATOR (Name and address of the organization preparing the document. A DRDC Centre sponsoring a contractor's report, or tasking agency, is entered in Section 8.) International Safety Research 38 Colonnade Road North Ottawa, (ON) Canada	2a. SECURITY MARKING (Overall security marking of the document including special supplemental markings if applicable.) CAN UNCLASSIFIED 2b. CONTROLLED GOODS NON-CONTROLLED GOODS DMC A

3. TITLE (The document title and sub-title as indicated on the title page.)

TYCHE 3.2 Upgrade

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

Watson, C.

5. DATE OF PUBLICATION (Month and year of publication of document.) May 2018	6a. NO. OF PAGES (Total pages, including Annexes, excluding DCD, covering and verso pages.) 30	6b. NO. OF REFS (Total references cited.) 12

7. DOCUMENT CATEGORY (e.g., Scientific Report, Contract Report, Scientific Letter.)

Contract Report

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

DRDC - Ottawa Research Centre
Defence Research and Development Canada
3701 Carling Avenue
Ottawa, Ontario K1A 0Z4
Canada

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.) W7714-156105	9b. CONTRACT NO. (If appropriate, the applicable number under which the document was written.)
10a. DRDC PUBLICATION NUMBER (The official document number by which the document is identified by the originating activity. This number must be unique to this document.) DRDC-RDDC-2018-C095	10b. OTHER DOCUMENT NO(s). (Any other numbers which may be assigned this document either by the originator or by the sponsor.)

11a. FUTURE DISTRIBUTION WITHIN CANADA (Approval for further dissemination of the document. Security classification must also be considered.)

Public release

11b. FUTURE DISTRIBUTION OUTSIDE CANADA (Approval for further dissemination of the document. Security classification must also be considered.)

12. KEYWORDS, DESCRIPTORS or IDENTIFIERS (Use semi-colon as a delimiter.)

Capability Based Planning; Force Structure Analysis; Monte Carlo; High Performance Computing; Parallel Processing; Tyche

13. ABSTRACT/RÉSUMÉ (When available in the document, the French version of the abstract must be included here.)

Abstract

Tyche is a Monte Carlo discrete-event simulation application tool developed by Defence Research and Development Canada's Centre for Operational Research and Analysis (DRDC CORA) to support the Canadian Armed Forces in conducting capability-based planning for force structure analysis. As part of a phased upgrade approach to enable the tool to handle medium-to-large-scale, multi-objective optimization problems, DRDC CORA established a requirement to transition the Tyche application to a fully-supported, modern programming language and perform significant upgrades.

Tyche 3.2 is a natural extension to Tyche 3.1. In Tyche 3.1 flexibility was added to allow a user to select a time scale appropriate to the problem at hand, and the application was readied for high performance computing environments (HPC) through the implementation of fractional simulations. In Tyche 3.2, first-use defects of Tyche 3.1 were rectified to improve the software reliability, performance benchmarking was identified, and the documentation was given a refresh.

Résumé

Tyche est un outil de l’application de simulation à événements discrets de Monte Carlo développé par le Centre d’analyse et de recherche opérationnelle de Recherche et développement pour la défense Canada (CARO RDDC) afin d’aider les Forces armées canadiennes à effectuer la planification fondée sur les capacités pour l’analyse de la structure des forces. Dans le cadre d’une approche de modernisation graduelle permettant à l’outil de traiter des problèmes d’optimisation multi objectif de moyenne à grande envergure, CARO RDDC a établi le besoin de transformer l’application Tyche en un langage de programmation moderne avec soutien complet et d’effectuer des mises à niveau importantes.

Tyche 3.2 est une suite logique de Tyche 3.1. On a ajouté de la souplesse à Tyche 3.1 pour permettre à un utilisateur de sélectionner l’échelle temporelle appropriée au problème à l’étude, et l’application a été préparée pour les environnements de calcul de haute performance (CHP) par la mise en place de simulations fractionnelles. Dans Tyche 3.2, les défauts à la première utilisation de Tyche 3.1 ont été corrigés pour améliorer la fiabilité du logiciel, des tests de performance ont été déterminés, et la documentation a été mise à jour.