TYCHE 3.1 UPGRADES CONTRACTOR REPORT

W7714-156105 T009
ISR Report 6078-01-01
Version 3.0
1 May 2017

Presented to:

Cheryl Eisler and Fred Ma
Strategic Planning Operational Research Team
DGSTCO, Centre for Operational Research and Analysis
Defence Research and Development Canada

Prepared by:

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

DRDC-RDDC-2017-C099

QUALITY ASSURANCE AND VERSION TRACKING

Authorization

Title	**TYCHE 3.1 Upgrades Contractor Report**	
Report number	6078-01-01	
Version	3.0	Signature
Prepared by	Chad Watson	
Reviewed by	Devin Duncan	
Approved by	Ian Becking	
Approved for Corporate Release by	Ian Becking	

Version Tracking

Ver.	Action	By	Date
1.0	Release to Client	McCall	28 Mar 2017
2.0	Release to Client	Becking	31 Mar 2017
3.0	Edits for Internal Publication	Eisler	18 Apr 2017

TABLE OF CONTENTS

1. INTRODUCTION

Tyche is a Monte Carlo discrete-event simulation software tool that was originally developed by Defence Research and Development Canada - Centre for Operational Research and Analysis (DRDC CORA) to allow the Royal Canadian Navy to conduct capability-based planning 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), Tyche was recently converted to fully-supported, modern programming language (Microsoft Visual C#.NET) with a number of functional improvements (Tyche 3.0).

Following this conversion, work was needed to implement new features in the Graphical User Interface (GUI), fix bugs, and add the ability to run in parallel on multiple CPU cores and on high-performance computing (HPC) systems. A project was undertaken to implement these changes in Tyche 3.1.

1.1 Purpose

The purpose of this report is to provide an overview of the work performed in the upgrade to Tyche 3.1, discuss issues that arose, and present recommendations for furthering the development of Tyche.

At a high level, the following work was performed:

- Brought the new user-selectable time scales from the Simulation Engine into the GUI, Simulation Manager, and input/output files;
- Fixed bugs with assignment and rescheduling of assets;
- Enhanced the Operational Schedule (OpSched) Viewer, and provided highlighting and single iteration viewing;
- Parallelized iteration execution, resulting in a new simulation execution type, in preparation for multi-core runs;
- Readied Tyche for a high-performance computing (HPC) environment, resulting in a new simulation execution type;
- Aligned the User Guide content to the compiled help file content by switching to LaTeX authoring and publishing to .pdf and .chm; and
- Updated the TYCHE help documentation [1].

1.2 Timeline

The Tyche 3.1 upgrade project commenced October 3, 2016, provided a candidate release of 3.1 on March 20th, 2017 (as release 3.0.13) and was completed by March 31, 2017.

2. PROJECT DETAILS

2.1 SOW Tasks

The statement of work (SOW) was revised in February 2017, and listed the following tasks:

6.1 The contractor must update the Tyche GUI and SM to support user-selectable timescales (already developed in the Simulation Engine (SE)). This includes any modifications required to the input and output file formats.

6.2 The contractor must perform the following bug fixes and minor improvements:

a. Confirm that the bumping (rescheduling) of assets is implemented correctly to ensure that there are i) no overlapping or incorrect assignment days and ii) travel times are reported correctly when as asset is bumped multiple times in a row. If the implementation is incorrect, the contractor will fix it, ensuring that the asset assignment procedure remains correct otherwise. Sample input/output data will be provided by the Technical Authority (TA) for testing purposes.

b. Trace and fix the registered array of “essential to” assets, such that co-dependent demand chains register the correct assets to each other.

c. Ensure that when bumping (rescheduling) assets from one event to another, all dependent assets are bumped as well. Incorrect asset registration may be related to 6.2 a and b.

d. When the OpSched Search function saves output to a text file, the search criteria must be written to the beginning of the file as a record of the parameters of the search.

e. In the OpSched Viewer, a button must be added to open a new window that allows the user to view one iteration’s worth of data in a text-based editor.

f. In the OpSched Viewer, when the user right-clicks on a scenario phase or an asset level, a white (if the background is black, or black if the background is white) bar must appear across both panes of the OpSched Viewer in the background of the event timeline. The bar must be the height of the event, so that the user can see where other assets/scenarios do or do not align with the currently selected event.

g. Additional minor bug fixes and/or code/GUI improvements as identified during the course of the task. This may include generating the content for the political risk Excel spreadsheet automatically from the input/output data (based on version of Microsoft Excel®).

6.3 The contractor must parallelize the Simulation Engine in the following manner to run in Windows 7, while making any subsequent necessary changes to the Simulation Manager to ensure continued functionality and compatibility:

a. Restructure the simulation iterations to generate values to populate the random number generator at the beginning of each run (e.g. for 1000 iterations, 1000 values will be generated initially and then each value will be used to seed the random number generator for a given iteration). The individual iterations will be subdivided as separate threads, where one simulation executable will manage the threads for its own iteration

processes. Each individual iteration will write its results to a separate file (so results will not be lost in case of errors), which will then be appended in order to create the final output file once the iteration is complete, and the individual files will be deleted. The contractor is suggested to propose alternative methodologies for parallelization that will help improve computational speed/performance, as the parallel version must not have more than a 10% increase in total CPU run time (if the individual iteration CPU run times were added serially, compared to the total simulation CPU run time before parallelization) allowing for minor overhead due to thread management. A sample input file 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.

b. The thread management system must take advantage of as many cores as are a) currently allotted by any HPC system, b) the maximum number of cores allowed to run on any node/blade/personal computer (PC) specified in the settings of the Tyche SM [2], and c) currently available on the HPC system/PC in use in a shared fashion between running versions of Tyche. For example, if there are 3 Tyche executables running on a PC with a maximum of 16 cores permitted to run, and there are 15 of 16 cores available on the machine (assuming one core is being used by another process), then each executable would receive access to 5 cores. The thread management system should also be designed for use in a Windows HPC Server 2008 environment across multiple nodes/blades. This will require efficient file and processor management to minimize the amount of inter-node/blade communication.

6.4 The contractor must create a compiled HTML help file (.chm) whose sub-topics must be linked to the context-sensitive help in the Tyche Version 3.1 Visual C#.NET code. The help file must utilize the table of contents structure and topic content in [2] and [3]; 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.

6.5 The contractor must produce an updated version of the help files, meaning that all text and figures must reflect the most recent version of the Tyche code provided to the contractor 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 will 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.

6.6 The contractor will document the results of their work in a report.

2.2 Task Assumptions

This task was started with the following assumptions:

- Technical:
  - Quality – the code uses standard coding practices, patterns and commenting
  - Test:
    - regression tests exist that prove the current feature set
    - unit tests exist that prove the implementations
    - A suitably configured VM will suffice as a test platform (Windows 7 licenses are readily available)
  - Technology – there are no underlying technologies which cannot be parallelized
  - Engineering – suitable test platforms are available
  - Training – A senior resource can operate the software reasonably well within a day
  - Integration – the existing code base can be integrated into the existing source code control repository and build system within a day.

- Programmatic:
  - Estimates:
    - Integration will take no more than a day for one person
    - Training will take no more than a day for each resource
    - The proposed estimates are in line with the customer's previous experience
    - Working knowledge of the current application is not required to accurately estimate the effort
  - Schedule:
    - Work Authorization will arrive no later than Sep 9th.
  - Contract Structure:
    - The contractor will be able to shift hours from one resource to another provided the overall dollar value of the task is not exceeded.
- Business:
  - Resources
    - Any subject matter experts that will be involved in the confirmation of functionality will be involved and available from the beginning of the contract until the end (with minor absences for vacation)
    - All government furnished information and equipment will be made available on the first day of the contract.

2.3 Work Undertaken

The following main development tasks occurred to produce Tyche version 3.1:

1. **Time Scales**. The contractor updated the Tyche GUI and SM to support user-selectable time scales that were already developed in the Simulation Engine (SE). This required modifications to the input and output file formats.

2. **Bug Fixes**. The contractor performed the following bug fixes and minor

improvements:

a. **Assignment Marks**. The bumping (rescheduling) of assets was found to be implemented correctly but in-correct assignment days were affecting the correct determination of a bump event. Fixes were applied to the calculation of assignment marks to mitigate overlapping assignments as well as to the lookup of travel distances that were returning zero travel time in some calculations.
b. **Essential To**. The registered array of “essential to" assets was traced, faults were found and fixed. Co-dependent demand chains register the correct assets to each other.
c. **Bump Dependent Assets**. Some of the errors in assignment marks were related to bumping. By the time the assignment marks were fixed, all cases of bumping were tested as well. When bumping (rescheduling) assets from one event to another, all dependent assets are bumped as well. As noted in section 3.3.5.5 of the Tyche documentation, the condition of “A non-supplier lingers in theatre" still exists but has low risk and low impact on likely Tyche models.
d. **OpSched Search**. When the OpSched Search function saves output to a text file, the search criteria is now written to the beginning of the file as a record of the parameters of the search.
e. **Single Iteration Viewer**. In the OpSched Viewer, a button was added that opens a new window that allows the user to view one iterations worth of data in a text-based editor.
f. **Highlighter Bar**. In the OpSched Viewer, when the user right-clicks on a scenario phase or an asset level, a white (if the background is black, or black if the background is white) appears across both panes of the OpSched Viewer in the background of the event timeline. The bar is the height of the event, so that the user can see where other assets/scenarios do or do not align with the currently selected event.

3. **Parallelization**. The concept for parallelization changed from the proposed thread management implementation to a fractional simulation (executable) implementation. The contractor parallelized the Simulation Engine by splitting the main simulation into fractional simulations, one simulation per iteration. A parallel simulation execution consists of three main steps: preparation, execute and merge. The preparation step splits a simulation into fractional simulations. The execution step runs each fractional simulation and produces fractional outputs. The last step is a merge step where all the fractional files are merged back into a standard TYO (and statistics calculated - if required). The implementation was tested on Windows 7 and 10. The Tyche SM used to process iterations only in serial, but now has two execution modes, serial and parallel. Refer to sections 3.4.2.1 and G.4.2 of the Tyche documentation. The parallelization involved:
    a. **Random Seed Generator Usage**. The usage of the random seed generator within Tyche was changed slightly. The random seed generator used to hand out random numbers as the simulation progressed from one iteration to another. Now, the simulation's random seed is used to generate values (an iteration seed) to populate the random number generator at the beginning of each run (e.g. for 1,000 iterations, 1,000 values will be generated initially and then each value will be used to seed the random number generator for a given iteration).
    b. **Fractional Simulations**. The individual iterations were subdivided into

separate single iteration simulations, called fractional simulations. When a simulation is executed in parallel, the Tyche SM manages a three step process of splitting the simulation in fractional simulations, executing the independent fractional simulations in parallel, and then re-constituting the output back into a final simulation output file. Each individual iteration (fractional simulation) writes its results to a separate file so results will not be lost in case of errors. The intermediate fractional simulation files are deleted in the final step, but a flag in the code may be set so that the intermediate files remain for debugging purposes.

c. **Performance**. In comparison to a serially executed simulation, the parallel simulation incurs some overhead in the start-up of a Tyche SE for each fractional simulation, as well as file reading and writing. A sanity test of the parallel version to check for a maximum increase in total CPU run time of no more than 10% was conducted but the results were not conclusive. The sanity test did not confirm being within 10%, but it did not point to any obvious source of the exceedance. Average iteration calculation times between serial and parallel simulations was relatively easy to extract, and there was no significant difference found. Test setup, environment and instrumentation is key to confirming the performance of the serial versus parallel simulation execution types.
d. **Thread Usage - HPC**. The fractional simulation concept was re-applied in the HPC environment in that the fractional simulations are handed out to the HPC environment using an HPC parametric task. The fractional simulation concept allows the HPC environment to take advantage of as many threads as are allotted by any HPC system. The HPC system requires an HPC job tailored to Tyche. Tyche makes the creation of a Tyche HPC job relatively easy by generating a skeleton Tyche HPC .xml file that can be uploaded to the HPC environment.
e. **Thread Usage - Personal Computer (PC)**. The maximum number of threads allotted to a parallel simulation execution is set in the Tyche Dashboard as Maximum Concurrent Simulations. This allows the user complete flexibility in how Tyche uses the resources on their PC. For example, if there is one (1) simulation with fifteen (15) iterations, in a Tyche SM dashboard with maximum concurrent simulations set to 15, and there are 16 threads available, when the simulation is run in parallel, fifteen executables will spawn and each will occupy a thread. When the same simulation is run in serial, one executable will spawn, and one thread will be used. *(Note about the example: one executable leads off the preparation of the simulation, then fifteen executables start, as there is one per iteration, and then there is a final executable to conduct the merge and statistics. In actuality, 2x the max number of concurrent allowed simulations (plus the prep and final tasks) will be queued. Enough to ensure that there is never a thread free/un-loaded. This is analogous to a wave concept, where the crest of the wave keeps ahead of the demand.)*

Note: The use of terminology related to threads versus cores and Tyche SEs should be clarified. The Tyche SE is single threaded. The original concept for parallelization involved making a single Tyche SE multi-threaded, but this concept was abandoned in the interest of error recovery. The implementation left the Tyche SE single threaded, but added logic to allow multiple Tyche SEs to conduct fractional pieces of a single

simulation. As a result, when terminology is used to the effect of “one executable will be spawned/queued and one thread will be used”, on a machine without hyper-threading, one core will be consumed. A machine with hyper-threading can run twice as many threads as there are cores. The implication of cores and threads is covered in more detail in section 4.2.2.1 of the Tyche documentation.

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 Work Authorization

Although unavoidable, the work took almost a month longer to start than originally expected due to contracting delays. This was mitigated by adding an additional part-time resource to the project and then increasing their hours to make up the schedule.

3.3 Regression Tests

Client-provided test files called “Target Exploratory – Sept 16old.tyi” and later “Target Exploratory – Sept 16old.tyo” were used extensively for “sanity” testing. The “Target Exploratory” files were excellent large scale tests, covering multiple aspects of the program, and well suited to error discovery but were too complex to use for diagnosis and tracking of individual heuristics. Smaller, more concise tests were written to test targeted portions of the functionality of Tyche. The level of effort involved in writing these tests was recovered from efficiencies gained in documentation writing. Conclusions and Future Work at (see Section 5) recommend a future maintenance upgrade to generate appropriate regression tests.

3.4 Training

Tyche is a non-trivial program and required more than a day for a senior resource to become proficient in its operation for the purposes of testing and debugging. Bug fixes took longer than expected due to this additional ramp-up time. The TA was heavily involved in identification and tracking effort of certain bugs. The TA’s involvement and responsiveness to questions was essential to the successful completion of the bug fixes.

The availability of targeted regressions tests (as mentioned above) would expedite developer training in the future, as the regression tests not only answer the question of what should be done, but also what boundaries the functionality was designed to exist within. Running a bank of regression tests is more efficient developer training than reviewing user guide documentation alongside the code.

3.5 HPC Test Platform

The HPC platform was not ready for testing the contractor’s HPC implementation. Additional testing will be required once it is available.

3.6 Parallelization Performance Degradation

The requirement of an execution run in “parallel” mode to have not more than a 10% increase in total CPU run time when compared to an execution in “serial” mode was

clear, but the method of testing was not clear. At contract end, there was still some discussion as to whether the final performance was acceptable due to ambiguity in the test method.

4. TASK COMMENTARY

4.1 Overview

Brief overviews of the outcome for each task performed (see Section 2.1 for reference) are provided in this section. Details on the work completed are contained in Section 2.3.

4.2 Task 6.1 – Time Scales

A previous contractor report [4] outlined the changes that were required to the GUI. Once the changes were applied, several fixes were required in the Simulation Engine to ensure Tyche continued to operate as expected in the [Days/Years] time scale. Limited testing was conducted on the alternate time scales of [Hours/Days] and [Minutes/Hours]. The [Days/Years] time scale was tested with the "Target Exploratory" test file.

4.3 Task 6.2 – Bug Fixes and Enhancements

Notwithstanding the lack of diagnostic tests, the bug fixes and enhancements were successful due to the conscientious and highly responsive feedback from the technical authority.

4.4 Task 6.3(a) – Parallelization

The original SOW recommended a thread-based solution but the final analysis favoured an executable-based solution. Refer to Section 2.3, item 3 for further details.

4.5 Task 6.3(b) – HPC

The HPC work was completed but not tested in the actual HPC environment. The executable-based parallelization solution mitigates much of the risk that Tyche will not work as expected when it is run in the HPC environment.

4.6 Tasks 6.4 & 6.5 – Documentation

LaTeX authoring allowed both the User Guide (.pdf) and compiled help file (.chm) documentation formats to be generated automatically. This added some complexity to the initial setup of the authoring environment and to the usage of the entire documentation toolchain,[1] but several benefits emerged.

There were some notable changes from the Tyche 3.0 documentation. In Tyche 3.0, the outlines of the Tyche 3.0 User Guide (.pdf) and the Tyche 3.0 compiled help file (.chm) were similar, but not the same. In Tyche 3.1, the .chm is sub-set of the .pdf and the subset of content is a 100% match to the .pdf content. Some Tyche.pdf content, such as the abstract is not included in Tyche.chm. Additionally, programmer content that was

[1] Refer to section G.5 in the Tyche documentation [1] for details on how to regenerate the documentation.

scattered throughout the user guide has now been collected into a single annex. Refer also to Annex A for relevant sections with updated User Guide content.

An install script was also provided, so that a separate installer can be created for documentation updates that are independent of the software installer. The script automatically updates the documentation files in the install folder, replacing the need for the user to copy the files in manually.

5. RECOMMENDATIONS FOR FUTURE WORK

5.1 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 and Gherkin are highly recommended). This will mitigate design fatigue as new features are added.

5.2 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 file before the simulation run.

2. A fully georeferenced model. Such a switch would support:
    a. A GIS service with Mil Std 2525B icons for data entry and playback. Data entry would be enhanced by being able to plot locations, specify transit routes and view them geo-spatially. Playback would enhance the modelling and stakeholder engagement by being able to simulate a playback of assets, dependent assets and co-dependent assets moving and bumping into and out of theatres.
    b. Same-Same theatre bumping. When an asset is in a theatre and is bumped, it returns to its base before re-deploying, even if it is being bumped from one event to another event in the same theatre. The artificiality of returning to home base on a same-same theatre bump is currently acceptable, but in future versions this could be improved.
    c. Inclusion of Theatre-to-Theatre distances so that Assets can transit directly from one location to another when rescheduled. In terms of data input, this is a major change significantly increasing the requirements for the user to define the values for each Theatre pair.

3. Implementation of attrition and/or capability degradation over time. This would enable the consideration of the possibility of accidents reducing availability of particular capabilities, or combat loss;

4. Refinement of the Asset selection algorithm to eliminate double-counting between marginal and required Capability Supply, as well as take into account layered capability contributions to the matched capability score. Both are necessary to ensure the best combination of Assets are selected, at all layers under consideration;

5. Implementation of a user-selectable assignment heuristics. This would allow the user to apply user-defined rules like the specialized lift Capability rule (refer to Section 3.4.1.6 of the Tyche documentation for further information) to specific Asset Types or specific Scenarios to better model force structure scheduler-specific exceptions and special cases;

6. Redesign of the event handler so that it allows for modelling of concurrent operations (two or more Scenarios provided with Capability from a given Asset at the same time), waypoints/forward basing, and alternative locations for events (such as Bases). This would create a flexible, joint tool that would be able to handle many common and unique military operational scheduling Scenarios; and

7. As matter of normal development, the user guide and help files will require continuous updating as the application continues to evolve.

ANNEX A. Tyche 3.1 Familiarization

The Tyche documentation [1] 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 3.1 work:

- 1.1 - Development
- 3.3.1 – Time Scales
- 3.4.1.4 – Random Seed
- 3.4.2.1 – Simulation Initialization
- 3.5.8 – Iteration Viewer
- 4.1.5 – Years/Days/Hours
- 4.1.7 – Scenario Type List
- 4.2.3 – Administering All Simulations
- 4.3 – HPC Simulations
- G.4.2 - Parallelization
- G.4.3 - HPC
- G.5 – Regenerate Documentation

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 files (electronic edition), Defence R&D Canada - CORA.

[2] EISLER, C. (2016), A User Guide to Tyche Version 3.0: Converting Tyche to a Modern Development Environment, DRDC CORA TM 2013-295.

[3] LIU, M.J. (2016), Tyche Version 3.0 Setup.exe installation file.

[4] RESTOULE, T., and Hossain, D., LeverageTek IT Solutions (2014), Timescale Enhancements in the Tyche Simulation Software: Programmer's Reference, DRDC CORA CR 2013-184.