SOFTWARE REQUIREMENTS SPECIFICATION

for

<Project>

Version 1.0 approved

Prepared by <author>

<Organization>

April 6, 2018

Contents

Revision History

Name	Date	Reason For Changes	Version
21	22	23	24
31	32	33	34

1 Introduction

1.1 Purpose

This system is designed to interface with SPECjbb 2015 and make it a more user friendly. The system will provide users with the ability to save an store various run settings for use with SPECjbb 2015.

1.2 Project Scope

<Provide a short description of the software being specified and its purpose, including relevant benefits, objectives, and goals. Relate the software to corporate goals or business strategies. If a separate vision and scope document is available, refer to it rather than duplicating its contents here.>

1.3 References

1. SPECjbb2015
2. Documentation

2 Overall Description

2.1 Product Perspective

<Describe the context and origin of the product being specified in this SRS. For example, state whether this product is a follow-on member of a product family, a replacement for certain existing systems, or a new, self-contained product. If the SRS defines a component of a larger system, relate the requirements of the larger system to the functionality of this software and identify interfaces between the two. A simple diagram that shows the major components of the overall system, subsystem interconnections, and external interfaces can be helpful.>

SPECtate (the System) is open source software (OSS) released under the MIT License. It is not a standalone system, it is designed to be interfaced with SPECjbb2015 (the Benchmarking System). It provides a simple mechanism for users of the Benchmarking System to manage various JVM and benchmark configurations. It is not required to use the Benchmarking System.

2.2 Product Functions

<Summarize the major functions the product must perform or must let the user perform. Details will be provided in Section 3, so only a high level summary (such as a bullet list) is needed here. Organize the functions to make them understandable to any reader of the SRS. A picture of the major groups of related requirements and how they relate, such as a top level data flow diagram or object class diagram, is often effective.>

1. Manage JVM and Benchmark System Configurations (Option Sets)

   These Option Sets are extended from the user-provided JVM and Benchmark System arguments. The System manages numerous Option Sets provided by the user, allowing the user to:

   a) Modify Option Sets as a configuration file

   b) Load/Select Option Sets (configuration

   c) Save Option Sets as configurations for a benchmark run

2. Execution based on Option Sets

3. User Interface

The GUI and CLI provide same functionality of the System, therefore users will be able to perform the same tasks whether they are using the command-line or the graphical interface.

a) (Ad-hoc?) CLI for management of configurations and invoking Benchmark System.

b) GUI for interactive management of configurations and invoking Benchmark System.

2.3 User Classes and Characteristics

<Identify the various user classes that you anticipate will use this product. User classes may be differentiated based on frequency of use, subset of product functions used, technical expertise, security or privilege levels, educational level, or experience. Describe the pertinent characteristics of each user class. Certain requirements may pertain only to certain user classes. Distinguish the most important user classes for this product from those who are less important to satisfy.>

2.4 Operating Environment

The application will always run under a Unix environment. Users may run the application remotely without access to a graphical user interface.

2.5 Design and Implementation Constraints

<Describe any items or issues that will limit the options available to the developers. These might include: corporate or regulatory policies; hardware limitations (timing requirements, memory requirements); interfaces to other applications; specific technologies, tools, and databases to be used; parallel operations; language requirements; communications protocols; security considerations; design conventions or programming standards (for example, if the customer's organization will be responsible for maintaining the delivered software).>

2.6 User Documentation

<List the user documentation components (such as user manuals, on-line help, and tutorials) that will be delivered along with the software. Identify any known user documentation delivery formats or standards.>

2.7 Assumptions and Dependencies

<List any assumed factors (as opposed to known facts) that could affect the requirements stated in the SRS. These could include third-party or commercial components that you

plan to use, issues around the development or operating environment, or constraints. The project could be affected if these assumptions are incorrect, are not shared, or change. Also identify any dependencies the project has on external factors, such as software components that you intend to reuse from another project, unless they are already documented elsewhere (for example, in the vision and scope document or the project plan).>

3 External Interface Requirements

3.1 User Interfaces

<Describe the logical characteristics of each interface between the software product and the users. This may include sample screen images, any GUI standards or product family style guides that are to be followed, screen layout constraints, standard buttons and functions (e.g., help) that will appear on every screen, keyboard shortcuts, error message display standards, and so on. Define the software components for which a user interface is needed. Details of the user interface design should be documented in a separate user interface specification.>

3.2 Hardware Interfaces

<Describe the logical and physical characteristics of each interface between the software product and the hardware components of the system. This may include the supported device types, the nature of the data and control interactions between the software and the hardware, and communication protocols to be used.>

3.3 Software Interfaces

<Describe the connections between this product and other specific software components (name and version), including databases, operating systems, tools, libraries, and integrated commercial components. Identify the data items or messages coming into the system and going out and describe the purpose of each. Describe the services needed and the nature of communications. Refer to documents that describe detailed application programming interface protocols. Identify data that will be shared across software components. If the data sharing mechanism must be implemented in a specific way (for example, use of a global data area in a multitasking operating system), specify this as an implementation constraint.>

3.4 Communications Interfaces

<Describe the requirements associated with any communications functions required by this product, including e-mail, web browser, network server communications protocols, electronic forms, and so on. Define any pertinent message formatting. Identify any communication standards that will be used, such as FTP or HTTP. Specify any communication security or encryption issues, data transfer rates, and synchronization mechanisms.>

4 System Features

4.1 System Feature 1

Option Sets

4.1.1 Description and Priority

Option Sets include all configuration information selected by the user for a single run of SPECjbb 2015.

4.1.2 Functional Requirements

1. The system shall allow the user to select any valid JVM on their machine. ($JAVA)
2. The system shall validate user provided JVM arguments prior to running SPEC. ($JAVAOPTS)
3. The system shall allow the user to select a run type of HBIR, HBIR_RT, PRESET, or LOADLEVEL
4. The system shall allow the user to select the number of runs to be executed.
5. The system shall allow the user to enable basic data collection. (vmstat -nt 1 > $RUN_NUM.vmstat.log &)
6. The system shall allow the user to select various SPECjbb options.
7. The system shall allow the user to input a run identifier ($TAG)
8. The system shall allow the user to input the number of Nodes.

4.2 System Feature 2

Option Set execution

4.2.1 Description and Priority

Each Option Set will be executed sequentially.

4.2.2 Functional Requirements

1. The system shall inform the user which group and run number is currently executing.
2. The system shall log all information into SUT.txt
3. The system shall create a results directory (named $TAG)
4. Tge system shall create the configuration file 'specjbb2015.props'
5. The system shall place the configuration file in the SPEC configuration directory (RC3/config/) prior to running SPEC.
6. The system shall place a copy of the configuration file in the results directory.
7. The system shall start the SPEC Controller. ($JAVA $JAVAOPTS -jar
8. The system shall execute each Node appropriately.
9. The system shall end all started processes once the Controller exits.
10. The system shall copy all results into the Results Directory (cp result/specjbb2015*/report*/*.html .)

4.3 System Feature 3

Option Set Node execution

4.3.1 Description and Priority

Users may select each Option Set to be run on a number of different nodes, which must be handled accordingly.

4.3.2 Functional Requirements

1. The system shall inform the user which Node number is being started.
2. The system shall execute the TxInjector.
3. The syste m shall execute Backend

4.4 System Feature 4

Save, Create, Delete, and Load Option Sets

4.4.1 Description and Priority

The user needs to be create, save, delete, and load Option Sets for quick execution.

4.4.2 Functional Requirements

1. Create Option Set
2. Load Option Set
3. Delete Option Set
4. Save Option Set

4.5 System Feature 5

User Interface

4.5.1 Description and Priority

The user needs to be able to visualize and organize Option Sets prior to running benchmarks.

4.5.2 Functional Requirements

1. Users need to be able to view available Option Sets
2. Users need to be able to arrange Option Sets in the order to be executed.
3. Users need to be able to indicate that they are ready to being benchmarking.

5 Other Nonfunctional Requirements

5.1 Performance Requirements

<If there are performance requirements for the product under various circumstances, state them here and explain their rationale, to help the developers understand the intent and make suitable design choices. Specify the timing relationships for real time systems. Make such requirements as specific as possible. You may need to state performance requirements for individual functional requirements or features.>

The System responds quickly in every action-response, both for the CLI and the GUI interfaces. There should be no substantially long delays while using the System, other than when the Benchmark System itself is running in the shell.

5.2 Safety Requirements

<Specify those requirements that are concerned with possible loss, damage, or harm that could result from the use of the product. Define any safeguards or actions that must be taken, as well as actions that must be prevented. Refer to any external policies or regulations that state safety issues that affect the product's design or use. Define any safety certifications that must be satisfied.>

The use of Benchmarking Software imposes inherent risk for the user. Since a benchmarking test can push the limits of the hardware and software running on a device, it may lead to varying types of software and/or hardware failure. The user should exercise caution when using benchmarking software, especially if used with overclocked hardware, or, hardware with a high-voltage power supply.

5.3 Security Requirements

<Specify any requirements regarding security or privacy issues surrounding use of the product or protection of the data used or created by the product. Define any user identity authentication requirements. Refer to any external policies or regulations containing security issues that affect the product. Define any security or privacy certifications that must be satisfied.>

Should we remove this section?

5.4 Software Quality Attributes

<Specify any additional quality characteristics for the product that will be important to either the customers or the developers. Some to consider are: adaptability, availability, correctness, flexibility, interoperability, maintainability, portability, reliability, reusability, robustness, testability, and usability. Write these to be specific, quantitative, and verifiable when possible. At the least, clarify the relative preferences for various attributes, such as ease of use over ease of learning.>

Correctness: The System is designed to meet the specifications described in this document, specifically under External Interface Requirements (Section 3) and System Features (Section 4). The System is correct if these requirements are satisified.

Testability: The original developers of the System use Continuous Integration (CI) and unit-tests, which will help provide quality software for both users of the software and subsequent developers.

Usability: Since the System is intended to make configuring the Benchmarking System easier, the usability of it is based on the usability of the interfaces and the the added simplification of invoking SPECjbb with the user input.

5.5 Business Rules

<List any operating principles about the product, such as which individuals or roles can perform which functions under specific circumstances. These are not functional requirements in themselves, but they may imply certain functional requirements to enforce the rules.>

The Benchmarking System does not restrict access based on user roles or permissions, and the System is designed similarly.