CMSC 447

Software Development Plan (SDP)

Revision 2

1 Scope

1.1 Identification

This Software Development Plan (SDP) serves to detail the plan for software creation and development for version 1.0.0. of the ROBBR system. No other releases of this system have been made, and no new releases are foreseen as of latest document revision.

1.2 System overview

The purpose of the ROBBR system is to allow a user to determine the best location for a new headquarters for some particular jewel thief syndicate. The system collects publicly available statistics about locations within the United States and the user provides criteria for filtering and ranking houses within a user-selected state. The system then generates a list of zip codes, weighed by their adherence to the user specified criteria, and displays it to the user. The ROBBR system is being developed for the customer who will be the sole sponsor, acquirer, and user. The ROBBR system will run on a bootable USB, which may be carried by the customer. The I am Root Software Engineering Team shall be the sole developers of the software, and any reference to the developer will refer to the I am Root Software Engineering Team. For privacy and security reasons, the customer is not identified.

1.3 Document overview

This SDP for the ROBBR system details the planning involved in the creation of this software system. This includes schedules, general and detailed development practices, decision-making rationale, and other organization-centric information. This document is only to be seen by customer and developer and is meant to explain all of the functions of the product.

1.4 Relationship to other documentation

Future documents include the Software Requirements Specification (SRS), Software Design Description (SDD), Software Test Description (STD), Software Test Report (STR), and Software User Manual (SUM). All documents refer to content in one-another as is relevant to the content they address. The SRS provides the program states and the requirements to be fulfilled by each state, along with their traceability, constraints, conditions for satisfaction, and other relevant information. The SDD details the architectural structure of the system and how it serves to facilitate the fulfillment of system requirements. The STP describes how the system is tested against the requirements, specifically their satisfaction conditions. The STR serves to show the state of requirement satisfaction. Finally, the software will be delivered with these documents, as well as an SUM that describes procedures for how to use the software.

2 Overview of required work

2.1 Requirements and constraints on the system and software to be developed

The ROBBR system is meant to fulfill the requirements detailed in the SRS, restated informally in the system overview (section 1.2 of this document). This plan follows the modified MIL

STD-498 documentation, provided by the Software Engineering course for which this system is being developed.

2.2 Position of the project in the system life cycle

The ROBBR system, at revision 2 of this document, is in the design stage of the software development life cycle. At this stage, the SRS and SDD documents are in revision 2 as well. This document still applies to version 1.0.0. of the system.

2.3 The selected program/acquisition strategy or any requirements or constraints on it

ROBBR shall be acquired and used solely by the customer. The customer shall acquire the system, as a program on a USB 3.0 flash drive with a bootable Linux operating system.

2.4 Requirements and constraints on project schedules and resources

The ROBBR must be delivered on a USB 3.0 flash drive with a working bootable Linux operating system preinstalled on it. A schedule constraint exists in the form of a project completion deadline. This is the project demonstration day at the close of this course semester (May 10th, 2018). Information about crime and housing is required for the program to display relevant data and this information may be provided outside of the ROBBR CSCI.

2.5 Other requirements and constraints, such as on project security, privacy, methods, standards, interdependencies in hardware and software development, etc.

The operating system must be loaded onto a internet-connected computer that is capable of booting an OS from USB 3.0. Speed-based constraints are flexible, however, a USB 3.0 enabled computer would significantly facilitate for satisfaction of these constraints.

3 Plans for performing general software development activities

3.1 Software development process

The developer will be using a variant of a standard waterfall method, which will include AGILE-like aspects to facilitate a more flexible system development environment. These include meetings to determine and justify progress, iterations and revisions of documents and deliverables created in previous SDLC stages, etcetera. This is as per course requirements and in accordance with the modified MIL STD-498 documentation provided by the Software Engineering course for which this system is being developed.

3.2 General plans for software development

3.2.1 Software development methods

The ROBBR software development will apply the following general methods:

1. Phase 1: Software Requirements. The project will follow the defined processes recorded in Section 5 to conduct software requirements analysis and develop the Software Requirements Specification (SRS) and preliminary User Documentation Description (UDD).

2. Phase 2: Software Design. The software architecture will consist of reusable modules for performing individual tasks. The project will follow the defined processes documented in Section 5. An automated tool will be used to create documentation relating to the code description and outline.
3. Phase 3: Software Unit Development, Test, and Integration. The project will develop new code (or modify reused code), unit test, integrate, and document software following the processes in Section 5. While reused code will not be expected to conform to a single coding standard, changed source code must be supplemented with sufficient new comments and standard code headers to meet commenting provisions of the coding standard and to promote understandability.
4. Phase 4: System Qualification Test and Delivery. The developer will conduct Qualification and Testing according to Qualification Test Plans and Procedures, and document results in a Test Report.
5. Phase 5: Support Installation and Use. The I am Root project team will provide support, solely to the customer, to software installation, acceptance testing, and user operation, only until the end of the semester (approximately May of 2018).

3.2.2 Standards for software products

1. Standards for formatting include the following:
    a. The indentation and spacing is dictated by the Python programming language, as part of language syntax.
    b. Capitalization is in adherence to the naming conventions defined below in section 4.2.2.d of this document.
    c. Order of information applies to header comments, and is as listed below in section 4.2.2.b of this document.
2. Each file shall have a standard header that defines the following information in the order defined below:
    a. File name
    b. File version number
    c. The developers
    d. A description of the file
    e. Required inputs and desired outputs
    f. Descriptions of all data used
    g. Description of all imports
    h. Any known bugs in the file
3. Algorithms, functions, and other programming design decisions are to be detailed with inline comments near the location of implementation.
4. Naming conventions for variables, parameters, procedures, files, etc. are as follows:
    a. Functions and methods will be done in camel case, with leading lowercase characters (ex: camelCase)
    b. Variables and parameters will be separated by underscores with leading lowercase characters (ex: underscore_case)
    c. Objects will be separated by underscores with leading uppercase characters

(ex: Upper_case)

5. While the system implementation will be primarily using object oriented programming concepts and structures, there shall be no restrictions on the use of programming language features. However, for the sake of simplicity, use of features that may seem unintuitive to persons following an OOP paradigm should be made clear in comments.

3.2.3 Reusable software products

3.2.3.1 Incorporating reusable software products

The project will utilize as much publically and freely available reusable software as seen fit by the developers.

3.2.3.2 Developing reusable software products

Emphasis will be placed on good software engineering practices such as encapsulation and heavy and descriptive in line comments to provide a basis for future development and internal unit reuse. However, no reusable software is to be released as a result of this project.

3.2.4 Handling of critical requirements

3.2.4.1 Safety assurance

Customer personal safety is unlikely to be a concern in this project. However, the developer makes no safety guarantees. The customer and all users of the software shall use it at their own discretion.

3.2.4.2 Security assurance

Software engineers on the I am Root team will only discuss information regarding the customer privately. Additionally, the entire program will be available solely on a bootable USB 3.0 flash drive given to the customer at termination of the project.

3.2.4.3 Privacy assurance

The program does not store previous searches and will be solely available on a bootable USB 3.0 flash drive. While the software is developed for a task of questionable legality, the ROBBR system guarantees no safety against tracking of online information access.

3.2.4.4 Assurance of other critical requirements

As critical requirements are identified, they will be documented. A plan to handle them will be devised by the developer, and the plan will be followed as development progresses.

3.2.5 Computer hardware resource utilization

Computer memory resources are allocated in order to store real-time and long-term location data. Real-time data is to be obtained from various sources on the internet. Utilization of these resources is not to be specifically tracked. Furthermore, the computer will need to meet several basic requirements outside of the scope of our programming which will be detailed in the SRS.

3.2.6 Access for acquirer review

After each phase of production, all relevant documents and code will be provided to the customer for review. After the review, customer and developer responses will be documented, and relevant changes will be made. Before the end of the next phase, improvements based upon customer input will be shown to the customer. New revisions to the documentation will be created as these changes and improvements are applied.

4 Plans for performing detailed software development activities

4.1 Project planning and oversight

4.1.1 Software development planning (covering updates to this plan)

Project planning will be recorded in this SDP, as well as future documentation. The planning details will be available in documentation relevant to the nature of the planning. All members of the development team will be responsible for the oversight of the documentation and software. Each member will be assigned relevant system design sections as seen fit.

4.1.2 CSCI test planning

Regression testing is planned and scheduled after each major component is individually integrated into the system.

4.1.3 System test planning

After integration of tested/corrected components into the system, the system is to be built and tested. At end of development, the system will be once again tested and modified for speed and functionality requirements.

4.1.4 Software installation planning

The software will be installed on a USB 3.0 flash drive, as part of a bootable Linux operating system residing on the drive.

4.1.5 Software transition planning

The contract for ROBBR will only be maintained until May 10th, 2018. No further development or transition will occur after this point.

4.1.6 Following and updating plans, including the intervals for management review

Management review will take place during meetings with customer as they are scheduled. Plan updates will occur after these meetings and as development progresses.

4.2 System requirements analysis

4.2.1 Analysis of user input

User input is to be specifically detailed in the SDP. This document is provided to the customer at every revision of the document. Any user confusion or clarification will be addressed in email between the customer and the designated point of contact (POC), as well as in scheduled meetings between the developer team and the customer.

4.2.2 Operational concept

Customer operational concept is to be parsed into a set of requirements, available in the SRS. As changes have been proposed by the customer, the developer will discuss and update documentation as necessary, and provide the customer with the updated documentation.

4.2.3 System requirements
Requirements for the system will stem from the developers' understanding of user requirements, and will be created and made traceable from each developer as seen fit. These requirements will be mentioned in the SRS, and will be addressed in the SDD.

4.3 Software design

4.3.1 CSCI-wide design decisions
Design decisions will be made by the team during meetings, as a selection from a number of suggested decisions. The overall system-wide design decisions shall be made by the development team based on the design requirements given by the customer, however the customer will not be concerned with these decision.

4.3.2 CSCI architectural design
The architectural design is decided upon by the developer after having gone over requirements and what is needed to fulfill them. As these are decided, the project structure should take form, and the project may be split into individual components that provide functionality.

4.3.3 CSCI detailed design
Detailed design decisions are left to individual developers working on their assigned tasks. As long as the decision yields a unit that is safe for integration, and does not create any CSCI-level issues, there is no group decision making to be done. Decisions and their rationale shall be documented in commentary (as mentioned in section 4.2.2.C of this document)

4.4 Software implementation and unit testing

4.4.1 Software implementation
The ROBBR software will be written primarily in the Python programming language. Currently, the Python Codecs, Time, Requests libraries are being used for downloading real-time info, and BeautifulSoup is being used to parse that info. The AppJar Python GUI library is used for the user interface driver portion of the program.

4.4.2 Preparing for unit testing
The software is to be broken into components that may be tested separately. Unit tests will be written alongside code units, and tests will be run at time of completion of each code unit.

4.4.3 Performing unit testing
For each ROBBR version modification, the affected software units will be tested by the programmer. Each programmer will run unit tests against modified portions of code, and record the failing results to reference when fixing the code.

4.4.4 Revision and retesting
Based on the results of unit testing, any necessary revisions to the code will be made by the developer in reference to the failure state of the unit test. The software unit will be rewritten to

accommodate any failure states. It will then be retested at a code unit level. The software documentation will be updated as needed. This procedure will be repeated until all test cases have been satisfied. If a component sees failure despite seemingly having passed the unit test, another developer may attempt to rewrite the unit test so as to ameliorate the weakness in functionality.

4.4.5 Analyzing and recording unit test results

The developer will evaluate and approve the results of unit testing. The failing test results and failure conditions will be logged and kept track of for later use.

4.5 Unit integration and testing

4.5.1 Preparing for unit integration and testing

Team members will prepare integration test cases and perform related integration testing. For each component, test cases will be developed using functional iterative testing methods. As functionality is implemented within the component and the system, the test cases will reflect new functionality.

4.5.2 Performing unit integration and testing

Upon approval by the I am Root software development team, the fixed code units will be integrated into the test version of the build, which will then undergo test cases. The test failure results will be recorded, and the failure will be traced down to individual code units.

4.5.3 Revision and retesting

Problems detected during integration testing will be recorded. Necessary revisions are performed by the writer of the most probably failing code units, and the writers of the unit tests against those units. After revisions are made, the software will be retested and the software documentation will be updated. This procedure will be repeated until all component and system test cases have been satisfied and up to delivery of the software.

4.5.4 Analyzing and recording unit integration and test results

The developer will keep track of all integration and unit test failures in a single document as a hierarchy of what code unit functionality (or lack thereof) may be affecting other units.

4.6 Preparing for software use

4.6.1 Preparing the executable software

A Linux distribution will be installed onto a USB 3.0 flash drive. The program, the python environment, and any dependencies will be installed onto this Linux installation. All necessary files will be loaded onto the flash drive before the USB 3.0 flash drive is delivered for the customer, ready for use.

4.6.2 Preparing version descriptions for user sites

Currently a single full release is planned for the ROBBR CSCI: version 1.0.0. All existing and future descriptions of this software system describe this release.

4.6.3 Preparing user manuals

A single Software User Manual (SUM) is to be prepared. The SUM will reflect instructions on how to perform according to the requirements. The SUM is to be written as soon as possible, and is to be finalized at the conclusion of the software development life cycle, as testing is concluded.

4.6.4 Installation at user sites
During the testing phase, the system will be loaded and tested on the USB 3.0 flash drive, in order to best prepare the software for its use environment. At the conclusion of this software development life cycle, the USB 3.0 flash drive will be delivered to the user, along with the SUM. No further installation or maintenance will be performed.

4.7 Software product evaluation

4.7.1 In-process and final software product evaluations
Final software evaluations shall be provided by the customer and given to the professor of the course to determine our final grade. As the developer provides documentation and system progress to the customer, the in-process evaluations will take place.

4.7.2 Software product evaluation records, including items to be recorded
Evaluation records shall be provided by the customer in the form of email feedback. In the case of face-to-face meetings a summary of agreed upon feedback shall be emailed to the customer for verification and approval.

4.7.3 Independence in software product evaluation
Software will be evaluated by the customer, not an independent entity. Any problems will be reported directly to the development team.

4.8 Software quality assurance

4.8.1 Software quality assurance evaluations
After completion of each phase the customer will be shown the project completed so far. This will allow the customer to give direct feedback on the program to the development team.

4.8.2 Software quality assurance records, including items to be recorded
Records will be emailed to the customer when significant updates/revisions are made.

4.9 Other software development activities

4.9.1 Risk management, including known risks and corresponding strategies
Risks will be evaluated for consideration based likelihood and consequences. Current risks are as follows:

- ❑ Missing documentation deliverable dates
- ❑ Lack of experience with skills necessary for system implementation
- ❑ Changing customer demands

4.9.2 Software management indicators, including indicators to be used
A document of reference material includes temporary implementation notes, memos and other momentarily valuable information. Problems regarding implementation are reported in this

document as they arise. Other than this, no indicators exist. Changing customer demands will be included in new documentation revisions and addressed in software development as they are introduced.

4.9.3 Security and privacy

No security or privacy considerations exist at this time. The system is designed to have security and privacy easily managed by the customer during time of use.

5 Project organization and resources

5.1 Project organization

Every member of the development team will work under the customer directly. A single point of contact (POC) will communicate with the customer directly. The communication will be visible to all members of the group. Members will discuss what tasks they work on before they start, and describe the work that has been completed after they've finished their work session.

5.2 Project resources

Personnel resources exist in the form of a predetermined group of students. The personnel currently comprises 6 students. No loss or gain of personnel is expected, as of revision 2 of this document. A work-pool exists, in the form of writing project documentation, writing portions of individual components of the system, creating unit/integration/system tests, setting up the environment in which the system runs, etc. The POC takes the responsibility for communicating with the customer directly, however no other roles have been specifically designated. People decide on what they will work on as it comes up. As of now, different members of the developer are working on different components and on different aspects of the documentation as they see fit.

6 Notes

6.1 Acronyms

SDD	Software Design Description
SDP	Software Development Plan
SRS	Software Requirements Specification
STD	Software Test Description
STR	Software Test Report
SUM	Software User Manual
POC	Point of Contact