|  |
| --- |
|  |
| Tunnel-K Software Development Plan (SDP) |
|  |
|  |
| **Josh Calahan, Wes Cothran, Chris Davis, Michael Lynch, Brian Pittman** |
| **11/20/2011** |

**Approvals**

**\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ \_\_\_\_\_\_\_\_\_\_\_**

Instructor Approval: Dr. Jeffrey Kulick Date

**\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ \_\_\_\_\_\_\_\_\_\_\_**

Team Member: Josh Calahan Date

**\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ \_\_\_\_\_\_\_\_\_\_\_**

Team Member: Wes Cothran Date

**\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ \_\_\_\_\_\_\_\_\_\_\_**

Team Member: Chris Davis Date

**\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ \_\_\_\_\_\_\_\_\_\_\_**

Team Member: Michael Lynch Date

**\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ \_\_\_\_\_\_\_\_\_\_\_**

Team Member: Brian Pittman Date

**Revisions**

|  |  |  |
| --- | --- | --- |
| **Revision Number** | **Date** | **Description** |
| Initial | 11/20/11 | Original Draft |

Contents

[1) Scope 8](#_Toc309989619)

[1.1) Identification 8](#_Toc309989620)

[1.2) System overview 8](#_Toc309989621)

[1.3) Document overview 8](#_Toc309989622)

[1.4) Relationship to other plans 8](#_Toc309989623)

[2) Referenced documents 8](#_Toc309989624)

[3) Overview of required work 9](#_Toc309989625)

[4) Plans for performing general software development activities 9](#_Toc309989626)

[4.1) Software development process 9](#_Toc309989627)

[4.2) General plans for software development 9](#_Toc309989628)

[4.2.1) Software development methods 9](#_Toc309989629)

[4.2.2) Standards and practices for software products 9](#_Toc309989630)

[4.2.3) Traceability 9](#_Toc309989631)

[4.2.4) Reusable software products 9](#_Toc309989632)

[4.2.5) Handling of critical requirements 10](#_Toc309989633)

[4.2.6) Computer hardware resource utilization 10](#_Toc309989634)

[4.2.7) Recording rationale 10](#_Toc309989635)

[4.2.8) Access for acquirer review 10](#_Toc309989636)

[5) Plans for performing detailed software development activities 10](#_Toc309989637)

[5.1) Project planning and oversight 10](#_Toc309989638)

[5.1.1) Software development planning (covering updates to this plan) 10](#_Toc309989639)

[5.1.2) Software item test planning 10](#_Toc309989640)

[5.1.3) System test planning 11](#_Toc309989641)

[5.1.4) Software installation planning 11](#_Toc309989642)

[5.1.5) Software transition planning 11](#_Toc309989643)

[5.1.6) Following and updating plans, including the intervals for management review 11](#_Toc309989644)

[5.2) Establishing a software development environment 11](#_Toc309989645)

[5.2.1) Software engineering environment 11](#_Toc309989646)

[5.2.2) Software test environment 11](#_Toc309989647)

[5.2.3) Software development library 11](#_Toc309989648)

[5.2.4) Software development files 11](#_Toc309989649)

[5.2.5) Non-deliverable software 11](#_Toc309989650)

[5.3) System requirements definition 11](#_Toc309989651)

[5.3.1) Analysis of user input 11](#_Toc309989652)

[5.3.2) Operational concept 12](#_Toc309989653)

[5.3.3) System requirements 12](#_Toc309989654)

[5.4) System design 12](#_Toc309989655)

[5.4.1) System-wide design decisions 12](#_Toc309989656)

[5.4.2) System architectural design 12](#_Toc309989657)

[5.5) Software requirements definition 12](#_Toc309989658)

[5.6) Software design 13](#_Toc309989659)

[5.6.1) Software item-wide design decisions 13](#_Toc309989660)

[5.6.2) Software item architectural design 13](#_Toc309989661)

[5.6.3) Software item detailed design 13](#_Toc309989662)

[5.7) Software implementation and unit testing 13](#_Toc309989663)

[5.7.1) Software implementation 13](#_Toc309989664)

[5.7.2) Preparing for unit testing 13](#_Toc309989665)

[5.7.3) Performing unit testing 13](#_Toc309989666)

[5.7.4) Revision and retesting 13](#_Toc309989667)

[5.7.5) Analyzing and recording unit test results 13](#_Toc309989668)

[5.8) Unit integration and testing 14](#_Toc309989669)

[5.8.1) Preparing for unit integration and testing 14](#_Toc309989670)

[5.8.2) Performing unit integration and testing 14](#_Toc309989671)

[5.8.3) Revision and retesting 14](#_Toc309989672)

[5.8.4) Analyzing and recording unit integration and test results 14](#_Toc309989673)

[5.9) Software item qualification testing 14](#_Toc309989674)

[5.9.1) Independence in software item qualification testing 14](#_Toc309989675)

[5.9.2) Testing on the target computer system 14](#_Toc309989676)

[5.9.3) Preparing for software item qualification testing 15](#_Toc309989677)

[5.9.4) Dry run of software item qualification testing 15](#_Toc309989678)

[5.9.5) Performing software item qualification testing 15](#_Toc309989679)

[5.9.6) Revision and retesting 15](#_Toc309989680)

[5.9.7) Analyzing and recording software item qualification test results 15](#_Toc309989681)

[5.10) Software/hardware item integration and testing 15](#_Toc309989682)

[5.11) System qualification testing 15](#_Toc309989683)

[5.12) Preparing for software use 15](#_Toc309989684)

[5.12.1) Preparing the executable software 15](#_Toc309989685)

[5.12.2) Preparing version descriptions for user sites 15](#_Toc309989686)

[5.12.3) Preparing user manuals 16](#_Toc309989687)

[5.12.4) Installation at user sites 16](#_Toc309989688)

[5.13) Preparing for software transition 16](#_Toc309989689)

[5.13.1) Preparing the executable software 16](#_Toc309989690)

[5.13.2) Preparing source files 16](#_Toc309989691)

[5.13.3) Preparing version descriptions for the maintenance site 16](#_Toc309989692)

[5.13.4) Preparing the “as built” software item design and other software maintenance information 16](#_Toc309989693)

[5.13.5) Updating the system design description 16](#_Toc309989694)

[5.13.6) Updating the software requirements specification 16](#_Toc309989695)

[5.13.7) Updating the system/subsystem specification 16](#_Toc309989696)

[5.13.8) Preparing maintenance manuals 16](#_Toc309989697)

[5.13.9) Transition to the designated maintenance site 16](#_Toc309989698)

[5.14) Software configuration management 16](#_Toc309989699)

[5.14.1) Configuration identification 16](#_Toc309989700)

[5.14.2) Configuration control 17](#_Toc309989701)

[5.14.3) Configuration status accounting 17](#_Toc309989702)

[5.14.4) Configuration audits 17](#_Toc309989703)

[5.14.5) Packaging, storage, handling, and delivery 17](#_Toc309989704)

[5.15) Software product evaluation 17](#_Toc309989705)

[5.16) Software quality assurance 17](#_Toc309989706)

[5.17) Corrective action 17](#_Toc309989707)

[5.17.1)Problem/change reports, including items to be recorded 17](#_Toc309989708)

[5.17.2) Corrective action system 17](#_Toc309989709)

[5.18) Joint technical and management reviews 17](#_Toc309989710)

[5.18.1) Joint technical reviews, including a proposed set of reviews 17](#_Toc309989711)

[5.18.2) Joint management reviews, including a proposed set of reviews 18](#_Toc309989712)

[5.19) Risk management 18](#_Toc309989713)

[5.20) Software management indicators 18](#_Toc309989714)

[5.21) Administrative security and privacy protection 18](#_Toc309989715)

[5.22) Managing subcontractors 18](#_Toc309989716)

[5.23) Interfacing with software IV&V agents 18](#_Toc309989717)

[5.24) Coordinating with associate developers 18](#_Toc309989718)

[5.25) Project process improvement 18](#_Toc309989719)

[6) Schedules and activity network 18](#_Toc309989720)

[7) Project organization and resources 19](#_Toc309989721)

[7.1) Project organization 19](#_Toc309989722)

[7.2) Project resources 19](#_Toc309989723)

[8) Notes 19](#_Toc309989724)

[A) Annexes 19](#_Toc309989725)

## 1) Scope

## 1.1) Identification

Tunnel-K is an effort by graduate software engineering students at the University of Alabama in Huntsville enrolled in a two-semester Software Engineering Studio course (CPE656 and CPE658). The endeavor is aimed at building a small-scale wind tunnel and associated software systems and is intended to be used by science museums, schools, etc. for educational purposes. The work is being done in association with the Hands-On Science Center (HOSC) in Tullahoma, TN, and the team will draw upon their professional experience working as software engineers at Arnold Engineering Development Center (AEDC), Arnold AFB, TN. AEDC is the most advanced and largest complex of flight simulation test facilities in the world.

## 1.2) System overview

The overall Tunnel-K system consists of a wind tunnel structure along with associated computer hardware and software, wiring, sensors, motors, fans, power supplies, etc. used for controlling and monitoring the operation of the tunnel. Additionally, a two-dimensional flow solver application suite will provide the opportunity for experimentation with various shapes in a virtual wind tunnel environment and graphically displayed mach and pressure gradients. The suite will also provide integration with the physical wind tunnel controls so that simulated conditions and be illustrated in the real world.

## 1.3) Document overview

This document addresses the plans for performing general and detailed software development activities.

## 1.4) Relationship to other plans

No other project management plans will be utilized for this development.

## 2) Referenced documents

* Tunnel-K Statement of Work (SOW)
* Tunnel-K Rough Order of Magnitude Estimate (ROM)
* Tunnel-K Software Requirements Specification (SRS)
* Tunnel-K System Test Plan (STP)
* Tunnel-K Google Code site, <http://code.google.com/p/tunnelk>
* Tunnel-K Google Groups site, <http://groups.google.com/group/tunnelk>
* Trial-Use Standard for Information Technology Software Life Cycle Processes Software Development Acquirer-Supplier Agreement, J-STD-016-1995
* UAH CPE656 Fall 2011 Course Syllabus, Dr. Jeffry Kulick
* Code Conventions for the Java Programming Language, <http://www.oracle.com/technetwork/java/codeconvtoc-136057.html>
* Code Style Guidelines for Contributors, (Google Android Style Guide) <http://source.android.com/source/code-style.html>
* Writing a Library for Arduino, <http://arduino.cc/en/Hacking/LibraryTutorial>

## 3) Overview of required work

The requirements and constraints for this effort are identified in the SRS. Documentation requirements and constraints are specified by J-STD-016-1995 as well as the UAH CPE656 course syllabus. The work encompasses the system life cycle from inception through deployment. Support will be provided for operations and maintenance at least through the end of the Software Engineering Studio course, anticipated to end in May of 2012. Free and open-source software (FOSS), APIs, etc. will be used whenever possible to ensure the cost of replicating the system remains as low as possible.

## 4) Plans for performing general software development activities

## 4.1) Software development process

An iterative software development process will be used whereby small pieces of functionality are rapidly designed, implemented, and tested prior to being integrated into the larger system. Numerous builds will, therefore, be made, each supporting the goal of implementing a given portion of functionality.

## 4.2) General plans for software development

## 4.2.1) Software development methods

The software development standard J-STD-016-1995 will be used as a basis for documenting the various work stages in this effort, and the process described in section 4.1 will serve as the primary software development method. Numerous tools will be used to complete the work, including, but not limited to, Google Code, Google Groups, e-mail, Eclipse, the Android operating system APIs, Android emulators, the Arduino development environment, git, ArgoUML, Microsoft Office, Inkscape, Paint.NET, and Twitter.

## 4.2.2) Standards and practices for software products

Developers on this effort will conform to the standards, best practices, and conventions of the programming language(s) used. See section 2, Referenced Documents, for specific standards and conventions documents.

## 4.2.3) Traceability

This subclause shall describe the approach to be followed for performing upward and downward traceability.

## 4.2.4) Reusable software products

The process for identifying, evaluating, and incorporating reusable software products will entail evaluating by engineering judgment the openness of the products, the ease of integrating into the Tunnel-K system, and value of reusing a given product. Products expected to be reused include a 2D flow solver application written by Jim Masters (a consultant to the project), and the Arduino Ethernet library. Both are FOSS, and the benefit of incorporating the existing code far outweighs any drawbacks that may exist. No restrictions exist for the use of either product.

## 4.2.5) Handling of critical requirements

Critical requirements will be identified and handled in close cooperation with the HOSC. Of primary concern is ensuring the safety of science museum patrons, as a majority of them are expected to be children. Privacy of users is also of the utmost concern, as functionality for sharing wind tunnel data and results via social networks will be included in the system. Major security issues are not anticipated given the environment in which the system will exist.

## 4.2.6) Computer hardware resource utilization

The allocation of computer hardware will be handled using engineering judgment. Due to the embedded nature of the physical wind tunnel, real-time loops must be implemented which meet the timing needs of the application. Additionally, display parameters must be updated under the time constraints specified in the SRD. Therefore, resources must be allocated for each function of the physical wind tunnel while still meeting timing requirements. This might require multiple processors in the real-time system.

## 4.2.7) Recording rationale

All key decisions on the project will be recorded using the Tunnel-K Google Groups discussion board or the Tunnel-K Google Code repository, issue tracker, or wiki.

## 4.2.8) Access for acquirer review

The Google Code and Google Groups websites are publically viewable, ensuring all stakeholders of this project have constant access to review software products and activities.

## 5) Plans for performing detailed software development activities

## 5.1) Project planning and oversight

## 5.1.1) Software development planning (covering updates to this plan)

Software development planning and oversight will be performed by the entire team, as well as the customer and course instructor. Deliverables and deadlines are set by the course, and the team’s weekly meetings will provide an opportunity to make the necessary plans to meet deadlines.

## 5.1.2) Software item test planning

All software will be actively unit tested during development, but further test planning will be driven by the course schedule and by decision made in group meetings.

## 5.1.3) System test planning

Full system testing will be planned during weekly group meetings and will be driven by the course schedule.

## 5.1.4) Software installation planning

Not applicable. The software will be installed prior to the end of the second term of the course.

## 5.1.5) Software transition planning

Ownership of the Tunnel-K system will be transferred at the end of the second term, in May 2012. This will take place in accordance with the schedule set forth by the course instructor.

## 5.1.6) Following and updating plans, including the intervals for management review

Management review will occur weekly during group meetings, status reports to the course instructor, and monthly status updates to the customer.

## 5.2) Establishing a software development environment

## 5.2.1) Software engineering environment

The software engineering environment will be assisted by use of the tools specified in section 4.2.1. The IDEs, version control systems, and other tools will be used by all developers on the project.

## 5.2.2) Software test environment

Testing will be accomplished using the tools specified in section 4.2.1.

## 5.2.3) Software development library

All code and libraries will be stored and maintained in the Google Code git repository.

## 5.2.4) Software development files

All code will be stored and maintained in the Google Code git repository.

## 5.2.5) Non-deliverable software

Not applicable. All software built will be delivered to the customer.

## 5.3) System requirements definition

## 5.3.1) Analysis of user input

Defining requirements for user input will be done using the team’s engineering judgment and customer input. Informal peer reviews will be used to coordinate decisions before formally documenting them. Customer feedback will be sought prior to finalizing analysis of user input.

## 5.3.2) Operational concept

The operational concept will be defined using the team’s engineering judgment and customer input. The team will draw upon experience from the operational concepts used in the test facilities at AEDC. While this project is merely for an exhibit at a science museum, it is intended that the final product have a look-and-feel which resembles a large-scale test facility. Informal peer reviews will be used to coordinate decisions before formally documenting them. Customer feedback will be sought prior to finalizing the concept.

## 5.3.3) System requirements

The system requirements will be defined using the team’s engineering judgment and customer input. Informal peer reviews will be used to coordinate decisions before formally documenting them. Customer and instructor feedback will be sought prior to finalizing requirements.

## 5.4) System design

## 5.4.1) System-wide design decisions

An iterative design approach will be used based upon building each subsystem individually, designing each interface to integrate them separately. The physical and virtual wind tunnels will be the first breakdown of subsystems, and their combining those subsystems will be the largest integration step. Informal peer reviews will be used to coordinate decisions before formally documenting them. Customer and instructor feedback will be sought prior to finalizing system-wide design decisions.

## 5.4.2) System architectural design

The system architectural design will take place early in the development to reduce risk for the integration of subsystems, as well as to ensure the project uses FOSS components, libraries, tools, etc. This design will largely be left to the team’s engineering judgment. Informal peer reviews will be used to coordinate decisions before formally documenting them.

## 5.5) Software requirements definition

Software requirements will be defined using J-STD-016-1995. Use cases diagrams, use cases, architectural diagrams, and other UML documentation will also be utilized to track requirements. Customer and instructor feedback will be sought prior to finalizing requirements.

## 5.6) Software design

## 5.6.1) Software item-wide design decisions

These design decisions will be made by the team using engineering judgment. UML will be used to model the software in order to refine the design on paper prior to making decisions.

## 5.6.2) Software item architectural design

UML will be utilized to design the architecture of the system, and constraints will be placed upon the architecture to ensure FOSS components and tools are favored over those that are propriety. Customer feedback will be sought prior to finalizing the architectural design.

## 5.6.3) Software item detailed design

UML will be utilized to model software items and to refine the detailed design prior to implementation. Individual software items may go completely through requirements to implementation and testing prior to finalizing a complete software design for the entire system. Software item detailed design decisions will be made entirely by the team.

## 5.7) Software implementation and unit testing

## 5.7.1) Software implementation

Software implementation will be accomplished using industry best-practices based on the experience of the team’s engineers. Tools specified in section 4.2.1 will be utilized.

## 5.7.2) Preparing for unit testing

See section 5.7.3.

## 5.7.3) Performing unit testing

When possible, automated unit testing will tools will be utilized in a test-driven development approach (TDD). In cases where a TDD approach is not useful, unit testing will be performed by the team’s engineers using industry best-practices and techniques drawn upon from the team’s professional experience. The Tunnel-K System Test Plan will define testing to be accomplished.

## 5.7.4) Revision and retesting

Defects will be documented using the Google Code issue tracker, and issues will be tracked in that tool to completion.

## 5.7.5) Analyzing and recording unit test results

For automated unit tests, no recording will take place, as defects are to be corrected whenever found per TDD best-practices. Defects found while unit testing other portions of the system will either be fixed immediately or placed into the Google Code issue tracker to be addressed at a later time.

## 5.8) Unit integration and testing

## 5.8.1) Preparing for unit integration and testing

Unit integration and testing will take place at the customer site or in an alternate location after transporting the existing wind tunnel structure. Individual team members responsible for the physical wind tunnel portion as well as the virtual wind tunnel will need to be present to ensure any defects can be understood and documented correctly.

## 5.8.2) Performing unit integration and testing

Unit and integration and testing will be performed with as many team members present as possible. Defects will be fixed immediately or documented in the Google Code issue tracker to be addressed at a later time.

## 5.8.3) Revision and retesting

Defects and revisions will be tracked in the Google Code issue tracker and addressed in order of priority of the issue and the availability of key personnel.

## 5.8.4) Analyzing and recording unit integration and test results

Defects and revisions will be tracked in the Google Code issue tracker and addressed in order of priority of the issue and the availability of key personnel. Analysis of the defects will be performed by the team members present for unit integration and testing activities.

## 5.9) Software item qualification testing

## 5.9.1) Independence in software item qualification testing

When possible, qualification testing will be performed with the customer and course instructor present, ensuring concurrence with qualification.

## 5.9.2) Testing on the target computer system

Testing must be performed on the appropriate computer system, not in an emulated or simulated environment. A unique embedded system is to be tested, requiring that the physical components driven by the system function properly. Additionally, the virtual wind tunnel may be installed on special-purpose hardware for user interface purposes, requiring it be tested on the appropriate system.

## 5.9.3) Preparing for software item qualification testing

As qualification testing is merely showing that requirements have been met, preparations for software item qualification testing will be integrated in the development approach.

## 5.9.4) Dry run of software item qualification testing

One or many dry runs of software item qualification testing will take place prior to the final test. Only team members are expected to be present for dry runs.

## 5.9.5) Performing software item qualification testing

The final software item qualification testing will be performed in a scheduled meeting with all project stakeholders. Transition/delivery will take place at the end of this meeting.

## 5.9.6) Revision and retesting

Any revisions or retesting required to be performed after the qualification testing meeting will take place after the second term of the course. Any such revision and retesting will be done in coordination with the course instructor, as the outcome may affect course grading. It is the intention of the Tunnel-K team to ensure full customer satisfaction, whether the work is completed fully during the course term or beyond.

## 5.9.7) Analyzing and recording software item qualification test results

Analysis and recording of software item qualification test results will occur formally during the qualification testing meeting. Signatures from all stakeholders on an approval form will show concurrence with the results.

## 5.10) Software/hardware item integration and testing

See section 5.7. The same approach will take place for software/hardware item integration and testing as is performed for software item integration and testing.

## 5.11) System qualification testing

See section 5.9. The same approach will take place for overall system qualification testing as is performed for software qualification testing.

## 5.12) Preparing for software use

## 5.12.1) Preparing the executable software

Binaries will be generated using the appropriate development environments.

## 5.12.2) Preparing version descriptions for user sites

When possible, version descriptions will be available through the user interface based on git the revision number at the time of a build. The embedded system being implemented may not be able to provide a user/administrator with a version number.

## 5.12.3) Preparing user manuals

No user manuals are anticipated to be generated in this effort.

## 5.12.4) Installation at user sites

Installation of both the physical and virtual wind tunnel will take place after the qualification testing meeting.

## 5.13) Preparing for software transition

## 5.13.1) Preparing the executable software

See section 5.12.1.

## 5.13.2) Preparing source files

All source code will be available on the Tunnel-K Google Code site.

## 5.13.3) Preparing version descriptions for the maintenance site

See section 5.12.2.

## 5.13.4) Preparing the “as built” software item design and other software maintenance information

All such information will be provided on the Tunnel-K Google Code site.

## 5.13.5) Updating the system design description

All such information will be provided on the Tunnel-K Google Code site.

## 5.13.6) Updating the software requirements specification

All such information will be provided on the Tunnel-K Google Code site.

## 5.13.7) Updating the system/subsystem specification

All such information will be provided on the Tunnel-K Google Code site.

## 5.13.8) Preparing maintenance manuals

No maintenance manuals are anticipated to be generated in this effort.

## 5.13.9) Transition to the designated maintenance site

Transition will take place after the qualification testing meeting.

## 5.14) Software configuration management

## 5.14.1) Configuration identification

All source code (prototype or otherwise), documentation, drawings, etc. will be maintained as configuration items.

## 5.14.2) Configuration control

Configuration control will be maintained using the Tunnel-K git repository via Google Code.

## 5.14.3) Configuration status accounting

As this effort involves a single installation of the system and the latest version of software will always be utilized, configuration status accounting will merely involve ensuring the latest version is installed on the hardware when delivered to the customer. This can be ensured by using git to ensure the latest source code is used in the final build.

## 5.14.4) Configuration audits

No configuration audits will be performed given the short time period and of this effort and that fact that a single installation is to be made.

## 5.14.5) Packaging, storage, handling, and delivery

Not applicable.

## 5.15) Software product evaluation

Software product evaluations are to be tailored out of the process used in this effort. The final qualification test and preparations leading up to it will serve this purpose, and agreement that the product covers all requirements will be reached during that test.

## 5.16) Software quality assurance

Software quality assurance processes are tailored out of this effort. Each developer in the team will be responsible for QA, and changes to the processes in place will be addressed during weekly meetings as needed.

## 5.17) Corrective action

## 5.17.1)Problem/change reports, including items to be recorded

Problem/change reports will be recorded using the issue tracker provided by the Tunnel-K Google Code site. All required fields in the issue tracker must be filled out. Issues will be addressed in order of priority and by the availability of key personnel.

## 5.17.2) Corrective action system

See section 5.17.1.

## 5.18) Joint technical and management reviews

## 5.18.1) Joint technical reviews, including a proposed set of reviews

Joint technical reviews will take place during weekly team meetings, including status updates involving the course instructor.

## 5.18.2) Joint management reviews, including a proposed set of reviews

Joint management reviews will take place during weekly team meetings, status updates involving the course instructor, and by e-mail or in person with the customer.

## 5.19) Risk management

Technical risks will be managed by a heavy prototyping stage during the first term of the course and beyond, as needed. Risks will also be managed by constant communication between the team, customer, and course instructor. As risks are identified, mitigation strategies will be developed by the team, and any resulting action will be immediately assigned.

## 5.20) Software management indicators

The primary indicators used to manage this software effort will be the time spent by each developer versus the estimates made at the beginning of the course. These indicators will be tracked and briefed during each weekly status update.

Need to decide which, if any, indicators we will collect and use in addition to those described above.

## 5.21) Administrative security and privacy protection

Not applicable.

## 5.22) Managing subcontractors

Not applicable.

## 5.23) Interfacing with software IV&V agents

Not applicable.

## 5.24) Coordinating with associate developers

Not applicable.

## 5.25) Project process improvement

Improvements to the processes used on the project will be addressed during weekly team meetings, and adjustments will be made to address any identified issues. Any change, if approved by the team, will result in a change to this document.

## 6) Schedules and activity network

Schedules will be documented on the Tunnel-K Google Code wiki, and changes will also be briefed during weekly Tunnel-K status updates.

## 7) Project organization and resources

## 7.1) Project organization

The project team will be self-organizing with no hierarchical structure. Work will be broken down by committee according to the strengths and expertise of each team member. The Hands-On Science Center customer and course instructor will both provide feedback to ensure the team delivers a useful exhibit.

## 7.2) Project resources

The project team is made up of five software engineers, and one consultant will assist in the integration of an existing software package. The team will share responsibility for all aspects of the project and will self-organize to ensure each team member is responsible for an equal share of work. Developers will primarily use their own personal equipment for this effort. The existing wind tunnel structure will be provided by the Hands-On Science Center, and discussions will take place as the design nears completion on which party will be responsible for providing funding for certain system components. It is anticipated that all materials will largely be at the expense of the team’s expense, but the Hands-On Science Center has expressed the possibility of providing funding for a tablet, PC, or similar equipment for use in the final product. Team member’s personal homes and workplaces will be utilized for all work, meetings, etc., and the Hands-On Science Center will be used for most customer meetings, formal reviews, tests, and, potentially, some build-up/integration.

## 8) Notes

This clause shall contain any general information that aids in understanding this document (e.g., background information, glossary, rationale). This clause shall include an alphabetical listing of all acronyms, abbreviations, and their meanings as used in this document and a list of any terms and definitions needed to understand this document.

## A) Annexes

Insert annexes as needed.