

Test Plan
For Activity Timer

Heather Hunting
October 17, 2002

1.0 PURPOSE

This document will outline the overall test plan for Activity Timer. It will also list the major features to be tested or not tested. Additional details about testing and tracking testing will also be provided.

2.0 OVERALL TESTING APPROACH

The main testing approach will be structural testing, on the program level and also each sub-level. Each module will be tested based on its design. Once several modules have been tested, they will be combined into a subsystem. This subsystem will be similarly tested, and the process will be repeated up through the full system level. To further test the system, the customer and possibly another person will perform beta tests on the system by using it as a typical user might. The system will also be run on several computer systems with various versions of the Windows operating system to ensure that it is functional.

3.0 FEATURES TO BE TESTED

All features will be tested using structural testing. Cases based on the underlying code will be formed to ensure all possible input types are tested and properly handled. The main features to be tested are:

- Planning activities – reading files from Activity Timer format, creating/editing files, saving files
- Importing activities – reading files from RapidOps format
- Timing activities – starting, stopping, pausing, resetting timer
- Viewing/Modifying Trial Data – deleting specified trials
- Viewing Time Summary Chart – clicking specific row in table to highlight the corresponding bars in the chart and vice versa
- Viewing/Modifying Activity Comparison Chart – adding/deleting activities, adding comments for each activity, saving changes

4.0 FEATURES NOT TO BE TESTED

While the basic interactions (program links and importing activities) between Activity Timer and RapidOps will be tested, none of the RapidOps code will be specifically tested for this project.

5.0 ENTRANCE AND EXIT CRITERIA

Each module will be ready for testing as soon as it is implemented – basically, testing will be continuous. The exit criteria will be determined by the test cases; when at least two of each type of test case has successfully been handled, the module will be deemed ready for integration. This will continue up to the program level, where a much fuller set of cases will be tested.

6.0 REGRESSION TESTING

The majority of modules will be independent. Dependencies will be tracked through design and code comments where possible. When a module is modified, it will be checked for dependencies; if any exist, at least two of the dependent modules will be retested, with a minimum of two of each type of test case for that module. Regression testing will be used, but not extensively.

7.0 TEST ENVIRONMENT

The main system used for testing will be a Dell Precision 610 with a Pentium II Xeon 450 MHz processor and 512 MB SDRAM. The operating system will be Windows 2000 SP 2, and testing will be done through Visual Basic 6.0. Secondary testing systems will be Dell computers with various versions of Windows, testing the executable program by itself.

8.0 CODE CONTROL

The major code control mechanism will be the Visual Basic IDE. If it is not apparent that a piece of code is being properly testing, it will be isolated in a new project and tested in that manner. Integration will be tested both within VB and in the executable itself.

9.0 TEST LOG, DEFECT REPORTING, AND TRACKING

Two logs will be maintained. One will be a test log; it will contain details of each test above the simplest modules (test type, date and time performed, results, etc). The other will be a defect log; it will contain details of each defect found (date and time discovered, method of discovery, expected action to be taken, etc). Each defect will be analyzed; if an apparent, quick solution (less than five lines of code need to be modified) exists, it will be implemented. Otherwise, it will be reported to the customer; if his feedback indicates that this is significant, more time will be spent looking for and implementing a solution. Once a solution has been implemented, regression testing will be used as described in Section 6.