

Software Users Manual

Vol. 9 - "tsWxGTUI" Toolkit

Rev. 0.0.0 Pre-Alpha)

Author(s): Richard S. Gordon



Author Copyrights & User Licenses for "tsWxGTUI_Py2x" & "tsWxGTUI_Py3x" Software & Documentation

- Copyright (c) 2007-2009 Frederick A. Kier & Richard S. Gordon, a.k.a. *TeamSTARS*. All rights reserved.
- Copyright (c) 2010-2015 Richard S. Gordon, a.k.a. *Software Gadgetry*. All rights reserved.
- GNU General Public License (GPL), Version 3, 29 June 2007
- GNU Free Documentation License (GFDL) 1.3, 3 November 2008

Third-Party Component Author Copyrights & User Licenses

- Attribution for third-party work directly or indirectly associated with the *TeamSTARS* "tsWxGTUI_PyVx" Toolkit are detailed in the "COPYRIGHT.txt", "LICENSE.txt" and "CREDITS.txt" files located in the directory named `./tsWxGTUI_PyVx/Documents`.

Draft

Contents

1	SCOPE (System Specification)	5
1.1	Identification (System Specification)	5
1.2	Purpose (System Specification)	8
1.3	Document Overview (System Specification)	9
2	SOFTWARE SUMMARY	21
2.1	Software Application	21
2.1.1	Capabilities	21
2.1.2	Operating Improvements	36
2.1.3	Benefits	36
2.2	Software Inventory	37
2.2.1	Manifest for Python-2.x	38
2.2.2	Manifest for Python-3.x	39
2.3	Software Environment	39
2.3.1	Computer Equipment	40
2.3.2	Communications Equipment	41
2.3.3	Other Software	41
2.3.4	Forms, Procedures, or Other Manual Operations	43
2.3.5	Other Facilities, Equipment, or Resources	44
2.4	Software Organization and Overview of Operation	44
2.4.1	Logical Components of the Software	45
2.4.2	Performance Characteristics	60
2.4.3	Relationship to Interfacing Systems	61
2.4.4	Supervisory Controls	77
2.5	Security and Privacy	78
2.6	Assistance and Problem Reporting	79
3	ACCESS TO THE SOFTWARE	81
3.1	First-time User of the Software	82
3.1.1	Equipment Familiarization	82
3.1.2	Access Control	91
3.1.3	Installation and Setup	91
3.2	Initiating a Session	92
3.2.1	Step-by-step procedures for beginning work	92
3.2.2	Options available	92
3.2.3	Checklist for problem determination	92
3.3	Stopping and Suspending Work	92
3.3.1	How the user can cease or interrupt use of the software	93
3.3.2	How to determine whether normal termination or cessation has occurred	93

4 PROCESSING REFERENCE GUIDE 95

4.1	Capabilities.....	95
4.2	Conventions.....	95
4.3	Processing Procedures.....	95
4.3.1	(Aspect of software use).....	96
4.4	Related Processing.....	96
4.5	Data Backup.....	96
4.6	Recovery from Errors, Malfunctions, and Emergencies.....	96
4.7	Messages.....	97
4.8	Quick-Reference Guide.....	97

5 NOTES 99

5.1	Use Case(s).....	99
5.1.1	Bootting.....	100
5.1.2	Shutdown.....	100
5.1.3	Operating System Administration.....	101
5.1.4	"tsWxGTUI" Toolkit Development.....	107
5.1.5	Python Language Administration.....	107
5.1.6	"tsWxGTUI" Toolkit Development.....	108
5.1.7	Python Application Development.....	108
5.1.8	Python Application Usage.....	108
5.1.9	System Troubleshooting & Maintenance.....	108
5.1.10	System Administrator.....	109
5.1.11	Software Engineer.....	110
5.1.12	System Operator.....	111
5.1.13	Field Service Personnel.....	112
5.2	Baseline Toolkit Development Platforms.....	113
5.2.1	Apple 27" iMac Desktop.....	114
5.2.2	Apple 17" MacBook Pro Laptop.....	118
5.2.3	Dell 15.6" Inspiron 7000 Laptop.....	123

6 DEFINITIONS, ABBREVIATIONS, AND ACRONYMS 125

7 APPENDIXES 147

8 APPENDIX A - BASELINE HOST COMPUTER PLATFORMS 149

8.1	Currently Used Platforms (Release Notes).....	149
8.1.1	Python 2x Configuration Spreadsheet.....	153
8.1.2	Python 3x Configuration Spreadsheet.....	154

8.2	Previously Used Platforms (Release Notes)	156
9	APPENDIX B - API RELATIONSHIP	159

9.1	Comparison of wxPython with tsWxGTUI (Development Plan)	159
-----	---	-----

10	APPENDIX C - DELIVERABLES	163
-----------	----------------------------------	------------

11	APPENDIX D - LOG FILES	169
-----------	-------------------------------	------------

11.1	Log File Examples	169
11.1.1	Bit-Mapped Image Tree (Splash Screen Files for Graphical Applications)	170
11.1.2	Logs Tree (Files for Non-Graphical Applications)	172
11.1.3	Logs Tree (Files for Graphical Applications)	182

Draft


1 SCOPE (System Specification)

Define the extent of the area or subject matter that something deals with or to which it is relevant.

- *Identification (System Specification)* (on page 5)
- *Purpose (System Specification)* (on page 8)
- *Document Overview (System Specification)* (on page 9)

1.1 Identification (System Specification)

This paragraph shall contain a full identification of the system and the software to which this document applies, including, as applicable, identification number(s), title(s), abbreviation(s), version number(s), and release number(s).

PRODUCT	IDENTIFICATION
Abbreviation	"tsWxGTUI"
Icon	
Name	<p>TeamSTARS "tsWxGTUI_PyVx" Toolkit</p> <p>Generic alias for:</p> <ul style="list-style-type: none"> ▪ TeamSTARS "tsWxGTUI_Py1x" Toolkit (reserved for Python 2x backport to legacy Python 1x) ▪ TeamSTARS "tsWxGTUI_Py2x" Toolkit (updated baseline for Python 2x) ▪ TeamSTARS "tsWxGTUI_Py3x" Toolkit (ports from Python 2x until Python 2x enters bug-fix only update stage then becoming updated baseline for Python 3x) ▪ TeamSTARS "tsWxGTUI_Py4x" Toolkit (reserved for Python 3x port to future Python 4x)
Title	Python 2.x & Python 3.x based Command Line Interface (CLI) Toolkit with "Curses" based, "wxPython"-style Graphical-Text User Interface (GUI) Toolkit
Identification Number	N/A
Version Number	0.0.0

Release Number	0.0.0 (pre-alpha)
Build Date	07/03/2015

Draft

<p>Usage Terms and Conditions --- Key Points</p>	<p>This is free and open source software. The <i>TeamSTARS</i> "tsWxGTUI_PyVx" Toolkit and its third-party components are distributed in the hope that they will be useful, but WITHOUT ANY WARRANTY; WITHOUT EVEN THE IMPLIED WARRANTY OF MERCHANTABILITY OR FITNESS FOR A PARTICULAR PURPOSE. IN NO EVENT SHALL THE COPYRIGHT HOLDER OR CONTRIBUTORS BE LIABLE FOR ANY DIRECT, INDIRECT, INCIDENTAL, SPECIAL, EXEMPLARY, OR CONSEQUENTIAL DAMAGES (INCLUDING, BUT NOT LIMITED TO, PROCUREMENT OF SUBSTITUTE GOODS OR SERVICES; LOSS OF USE, DATA, OR PROFITS; OR BUSINESS INTERRUPTION) HOWEVER CAUSED AND ON ANY THEORY OF LIABILITY, WHETHER IN CONTRACT, STRICT LIABILITY, OR TORT (INCLUDING NEGLIGENCE OR OTHERWISE) ARISING IN ANY WAY OUT OF THE USE OF THIS SOFTWARE, EVEN IF ADVISED OF THE POSSIBILITY OF SUCH DAMAGE.</p> <p>Please note the following:</p> <ol style="list-style-type: none"> 1 Each <i>TeamSTARS</i> "tsWxGTUI_PyVx" Toolkit distribution includes a root directory whose name ("tsWxGTUI_PyVx") has a suffix "_PyVx" that reflects the associated Python language syntax version: <ol style="list-style-type: none"> a) "tsWxGTUI_Py1x" - <i>Reserved</i> for Root of first generation syntax files and subdirectories for Python 1.0.0-1.6.1. There is currently no compelling need to justify the substantial effort to Backport from Python 2.x syntax, semantics and libraries. There would be obsolete syntax and semantic issues to be resolved. For example, issues associated with the importing of modules and data. There would be unimplemented library issues to be resolved. For example, Python 1.x supported only a Command Line Interface. It would be necessary to Backport the Python 2.x curses library in order to support a character-mode Graphical-style User Interface. b) "tsWxGTUI_Py2x" - Root of second generation syntax files and subdirectories for Python 2.0.0-2.7.9. c) "tsWxGTUI_Py3x" - Root of third generation syntax files and subdirectories for Python 3.0.0-3.4.3. d) "tsWxGTUI_Py4x" - <i>Reserved</i> for future Root of next generation syntax files and subdirectories for Python 4.x. 2 You can use, modify and redistribute the distribution only under the <i>Terms and Conditions</i> files provided in the ".tsWxGTUI_PyVx/Documents" sub-directory: <ol style="list-style-type: none"> a) "Copyright.txt" - Attribution for the principal Author(s) of intellectual property created specifically for the <i>TeamSTARS</i> "tsWxGTUI_PyVx" Toolkit. b) "Credits.txt" - Attribution for those third-party Author(s) whose intellectual property has been adapted for use in the <i>TeamSTARS</i> "tsWxGTUI_PyVx" Toolkit. c) "License.txt" - Statements of rights, obligations and limitations stipulated by the Authors of intellectual property included in the <i>TeamSTARS</i> "tsWxGTUI_PyVx" Toolkit. d) "Notices.txt" - Announcement calling the <i>TeamSTARS</i> "tsWxGTUI_PyVx" Toolkit recipient's attention to the applicable "Copyright.txt", "Credits.txt" and "License.txt" files.
--	---

1.2 Purpose (System Specification)

The *TeamSTARS* "tsWxGTUI_PyVx" Toolkit software provides building blocks and tools for the rapid prototyping and development of application programs suitable for embedded systems.

It and the hardware/software platform on which it has been installed, provide the means for developing, documenting, enhancing, operating, troubleshooting, maintaining and supporting the embedded system. Those means include the following software components:

- 1 Release Distributions** - The Toolkit distribution in one or more Python language generation-specific form(s).
 - a) *TeamSTARS* "tsWxGTUI_Py1x" Toolkit only for the first generation Python language 1.0.0-1.6.1 (reserved for future Python 2x back-port use)
 - b) *TeamSTARS* "tsWxGTUI_Py2x" Toolkit only for the second generation Python language 2.0.0-2.7.9
 - c) *TeamSTARS* "tsWxGTUI_Py3x" Toolkit only for the third generation Python language 3.0.0-3.4.3
 - d) *TeamSTARS* "tsWxGTUI_Py4x" Toolkit only for the fourth generation Python language 4.0.0 (reserved for future Python 3x port use)
 - e) *TeamSTARS* "tsWxGTUI_PyVx" Toolkit containing two or more of the above single generation Python language releases
- 2 Toolkit Components** - Toolkit building-block components are general-purpose, re-usable and enable the application developer to focus on the application specific functionality and not waste effort re-inventing and re-implementing the functionality typical of Command Line and Graphical User Interfaces. Components include:
 - a) **tsToolkitCLI** - Python-based toolkit for development of applications featuring a Command Line Interface (CLI).
 - b) **tsToolkitGUI** - Python and Curses-based toolkit for development of applications featuring a character-mode Graphical-style User Interface (GUI).
- 3 Toolkit Applications** - Applications typically feature "user-friendly" Command Line and/or Graphical-style User Interfaces that can be controlled locally or remotely. Commercial, industrial, medical and military embedded systems are customized and optimized for a specific use. Unlike their general-purpose desktop, laptop and workstation counterparts, they typically have limited, application-specific processing, memory, communication, input/output and file storage resources. Typical applications include:
 - a) **Automation** - The use of various control systems for operating equipment such as machinery, processes in factories, telephone networks, steering and stabilization of ships, aircraft and other applications with minimal or reduced human intervention.
 - b) **Communication** - The application of telecommunications technology for the transmission of data to, from, or between computers over dedicated or time shared hardware.

- c) **Control** - The application of one or more devices, to manage, command, direct or regulate the behavior of other device(s) or system(s). Industrial control systems, as used in industrial production, control equipment or machinery. In open loop control systems output is generated based only on inputs. In closed loop (feedback) control systems current output is taken into consideration and corrections are made based on feedback.
- d) **Diagnostic** - The application of technology to locate problems with software, hardware, or any combination there of in a system, or a network of systems. Diagnostics typically provide guidance to the user to solve issues.
- e) **Instrumentation** - The application of technology for the measurement and control of process variables within a production or manufacturing area. An instrument is a device that measures a physical quantity such as flow, temperature, level, distance, angle, or pressure. Instruments may be as simple as direct reading thermometers or may be complex multi-variable process analyzers. Instruments are often part of a control system in refineries, factories, and vehicles.
- f) **Simulation** - The application of technology for the imitation of the operation of a real-world process or system over time. The act of simulating something first requires that a model be developed; this model represents the key characteristics or behaviors/functions of the selected physical or abstract system or process. The model represents the system itself, whereas the simulation represents the operation of the system over time.

1.3 Document Overview (System Specification)

This paragraph shall summarize the purpose and contents of this document and shall describe any security or privacy considerations associated with its use.

The Introduction is one of a set of reference document volumes for individuals installing, developing, maintaining, troubleshooting and using the *TeamSTARS* "tsWxGTUI_PyVx" Toolkit and application programs developed with the Toolkit. It and the other documents are described below.

VOL	TITLE	CONTENTS
0	Announcement	This document alerts potential and existing users of the availability, capabilities, limitations and sources for the latest source code release and technical support.
1	Brochure	<p>From Wikipedia, the free encyclopedia:</p> <p>"A brochure (also referred to as a pamphlet) is a type of leaflet.</p> <p>Brochures are advertising pieces mainly used to introduce a company or organization, and inform about products and/or services to a target audience.</p> <p>Brochures are distributed by mail, handed personally or placed in brochure racks.</p> <p>The most common types of single-sheet brochures are the bi-fold (a single sheet printed on both sides and folded into halves) and the tri-</p>

		<p>fold (the same, but folded into thirds). A bi-fold brochure results in four panels (two panels on each side), while a tri-fold results in six panels (three panels on each side).</p> <p>Other folder arrangements are possible: the accordion or "Z-fold" method, the "C-fold" method, etc. Larger sheets, such as those with detailed maps or expansive photo spreads, are folded into four, five, or six panels. When two card fascia are affixed to the outer panels of the z-folded brochure, it is commonly known as a "Z-card".</p> <p>Booklet brochures are made of multiple sheets most often saddle stitched (stapled on the creased edge) or "perfect bound" like a paperback book, and result in eight panels or more.</p> <p>Brochures are often printed using four color process on thick gloss paper to give an initial impression of quality. Businesses may turn out small quantities of brochures on a computer printer or on a digital printer, but offset printing turns out higher quantities for less cost.</p> <p>Compared with a flyer or a handbill, a brochure usually uses higher-quality paper, more color, and is folded."</p> <p>Included are the following topics:</p> <ol style="list-style-type: none"> 1 Features <ol style="list-style-type: none"> a) Command Line Interface (CLI) b) Graphical-style User Interface (GUI) c) System Block Diagrams 2 Usage Applications <ol style="list-style-type: none"> a) Development Platforms b) Embedded Systems 3 Design Goals <p>What you can look forward to doing with the TeamSTARS "tsWxGTUI_PyVx" Toolkit?</p> <ol style="list-style-type: none"> a) Local Equipment Monitoring & Control b) Remote Equipment Monitoring & Control 4 Design Non-Goals <p>What the TeamSTARS"tsWxGTUI_PyVx" Toolkit is NOT ?</p> 5 Usage Terms & Conditions 6 SCREENSHOTS <ol style="list-style-type: none"> a) XTERM Terminal Emulator with 8-Color / 64-Color Pairs b) VT100 Terminal Emulator with 1-Color / 2-Color Pairs
2	Introduction	<p>This document orients the reader to the goals and non-goals for the "tsWxGTUI_PyVx" Toolkit. Included are the following topics:</p> <ol style="list-style-type: none"> 1 SCOPE (System Specification)

		<ul style="list-style-type: none"> a) Identification (System Specification) b) Purpose (System Specification) c) Document Overview (System Specification)
		2 SYSTEM OVERVIEW (tsWxGTUI Introduction) <ul style="list-style-type: none"> a) Purpose of System and Software b) General Nature of the System and Software c) History of System Development, Operation and Maintenance
		3 OPERATOR INTERFACE <ul style="list-style-type: none"> a) Command Line Interface (CLI) b) Graphical User Interface (GUI)
		4 APPLICATION PROGRAMMING INTERFACE <ul style="list-style-type: none"> a) Command Line Interface API b) Graphical User Interface API
		5 TOOLS & UTILITIES <ul style="list-style-type: none"> a) tsLinesOfCodeProjectMetrics b) tsPlatformQuery c) tStripComments d) tsStripLineNumbers e) tsTreeCopy f) tsTreeTrimLines
		6 DEFINITIONS, ABBREVIATIONS, AND ACRONYMS
		7 REFERENCED DOCUMENTS <ul style="list-style-type: none"> a) Project Documents b) Release Distribution Documents c) External Documents
		8 NOTES <ul style="list-style-type: none"> a) Operator Interface Technology
		9 APPENDIXES <ul style="list-style-type: none"> a) Appendix A - Baseline Host Computer Platforms b) Appendix B - API Relationship c) Appendix C - Deliverables d) Appendix D - Accomplishments (Draft) e) Appendix E - Inherited, Field-Proven Computer

		<p>Technology</p> <p>f) Appendix F - History of System Development, Operation, and Maintenance</p>
3	Terms & Conditions	<p>This document alerts potential users and reminds existing users to the rules which the user must agree to abide by in order to use, modify and redistribute the "tsWxGTUI_PyVx" Toolkit and/or any of its components.</p> <p>Included are the following topics:</p> <p>1 TERMS & CONDITIONS</p> <p>a) Copyright - Identifies the original author(s) of source code and documentation for building block libraries and application programs.</p> <p>b) License - Identifies the original author's rules for using, modifying and redistributing source code and documentation for building block libraries and application programs.</p> <p>c) Notices - Identifier placed on copies of the source code and documentation to inform the world of copyright ownership and the applicable license.</p> <p>d) Splash Screen Designer's Guide - Identifies the original author's personal guidelines for incorporating Copyright and License notices in Commnd Line Interface(s) and Graphical-style User Interface(s).</p>
4	Software Development Plan	<p>This document specifies a developer's plans for conducting a software development effort. The term "software development" is meant to include new development, modification, reuse, re-engineering, maintenance, and all other activities resulting in software products.</p> <p>The plan provides the acquirer insight into, and a tool for monitoring, the processes to be followed for software development, the methods to be used, the approach to be followed for each activity, and project schedules, organization, and resources.</p> <p>Included are the following topics:</p> <p>1 SCOPE (System Specification)</p> <p>2 DEFINITIONS, ABBREVIATIONS, AND ACRONYMS</p> <p>3 OVERVIEW OF REQUIRED WORK</p> <p>a) Purpose</p> <p>b) Requirements and Constraints</p> <p>c) Concept</p> <p>4 PLANS FOR PERFORMING GENERAL SOFTWARE DEVELOPMENT ACTIVITIES</p> <p>a) Software Development Process</p>

		<ul style="list-style-type: none"> b) General Plans for Software Development 5 PLANS FOR PERFORMING DETAILED SOFTWARE DEVELOPMENT ACTIVITIES <ul style="list-style-type: none"> a) Project Planning and Oversight b) Establishing A Software Development Environment c) System Requirements Analysis d) Software Design e) Software implementation And unit Test f) Unit integration And Testong g) CSCI Qualification Testing h) CSCI/HWCI Integration And Testing i) System Qualification Testing j) Preparing for Software Use k) Preparing for Software Transition l) Software Configuration Management m) Software Product Evaluation n) Software Quality Assurance o) Corrective Action p) Joint Technical and Management Reviews q) Other Software Development Activities 6 SCHEDULES AND ACTIVITY NETWORK 7 PROJECT ORGANIZATION AND RESOURCES <ul style="list-style-type: none"> a) Project organization b) Project Resources 8 NOTES 9 APPENDIXES 10 TO-DO-LIST <ul style="list-style-type: none"> a) New Features b) Modifications c) Troubleshooting d) Validation/Regression Test 11 SAMPLE SCREEN SHOTS
5	System Specification	This document specifies the required behavior of an engineering system. The documentation typically describes what is needed by the system user as well as requested properties of inputs and outputs (e.g.

	<p>of the software system).</p> <p>Included are the following topics:</p> <ol style="list-style-type: none"> 1 SCOPE (System Specification) 2 DEFINITIONS, ABBREVIATIONS, AND ACRONYMS 3 REQUIREMENTS 4 QUALIFICATION PROVISIONS <ol style="list-style-type: none"> a) Demonstration b) Test c) Analysis d) Inspection e) Special Qualification Methods 5 REQUIREMENTS TRACEABILITY 6 NOTES <ol style="list-style-type: none"> a) Partial List of Widget Toolkits b) Use Case(s) <ul style="list-style-type: none"> System Administrator Software Engineer System Operator 7 APPENDIXES 8 APPENDIX A - REQUIRED STATES AND MODES 9 APPENDIX B - SYSTEM CAPABILITY REQUIREMENTS 10 APPENDIX C - SYSTEM EXTERNAL INTERFACE REQUIREMENTS 11 APPENDIX D - SYSTEM INTERNAL INTERFACE REQUIREMENTS 12 APPENDIX E - SYSTEM INTERNAL DATA REQUIREMENTS 13 APPENDIX F - ADAPTATION REQUIREMENTS 14 APPENDIX G - SAFETY REQUIREMENTS 15 APPENDIX H - SECURITY AND PRIVACY REQUIREMENTS 16 APPENDIX I - SYSTEM ENVIRONMENT REQUIREMENTS 17 APPENDIX J - COMPUTER RESOURCE REQUIREMENTS
--	--

		<p>18 APPENDIX K - SYSTEM QUALITY FACTORS</p> <p>19 APPENDIX L - DESIGN AND CONSTRUCTION CONSTRAINTS</p> <p>20 APPENDIX M - PERSONNEL-RELATED REQUIREMENTS</p> <p>21 APPENDIX N - TRAINING-RELATED REQUIREMENTS</p> <p>22 APPENDIX O - LOGISTICS-RELATED REQUIREMENTS</p> <p>23 APPENDIX P - OTHER REQUIREMENTS</p> <p>24 APPENDIX Q - PACKAGING REQUIREMENTS</p> <p>25 APPENDIX R - PRECEDENCE AND CRITICALITY OF REQUIREMENTS</p>
6	Interface Requirements Specification	<p>This document specifies the requirements imposed on one or more systems, subsystems, Hardware Configuration Items (HWCIs), Computer Software Configuration Items (CSCIs), manual operations, or other system components to achieve one or more interfaces among these entities. An IRS can cover any number of interfaces.</p> <p>Included are the following topics:</p> <p>1 SCOPE (System Specification)</p> <p>2 DEFINITIONS, ABBREVIATIONS, AND ACRONYMS</p> <p>3 REQUIREMENTS</p> <p>a) Interface Identification and Diagrams</p> <p>b) (Project unique identifier of interface)</p> <p>c) Terminal Device Interface (TDI)</p> <p>d) Precedence and Criticality of Requirements</p> <p>4 QUALIFICATION PROVISIONS</p> <p>5 REQUIREMENTS TRACEABILITY</p> <p>6 NOTES</p> <p>7 APPENDIXES</p>
7	Software Requirements Specification	<p>This document specifies the requirements for a Computer Software Configuration Item (CSCI) and the methods to be used to ensure that each requirement has been met. Requirements pertaining to the CSCI's external interfaces may be presented in the SRS or in one or more Interface Requirements Specifications (IRSs).</p> <p>Included are the following topics:</p> <p>1 SCOPE (System Specification)</p> <p>2 DEFINITIONS, ABBREVIATIONS, AND ACRONYMS</p> <p>3 REQUIREMENTS</p>

		<ul style="list-style-type: none"> a) Required States and Modes b) CSCI Capability Requirements c) CSCI External Interface Requirements d) CSCI Internal Interface Requirements e) CSCI Internal Data Requirements f) Adaptation Requirements g) Safety Requirements h) Security and Privacy Requirements i) CSCI Environment Requirements j) Computer Resource Requirements k) Software Quality Factors l) Design and Construction Constraints m) Personnel-related Requirements n) Training-related Requirements o) Logistics-related Requirements p) Other Requirements q) Packaging Requirements r) Precedence and Criticality of Requirements <p>4 QUALIFICATION PROVISIONS</p> <p>5 REQUIREMENTS TRACEABILITY</p> <p>6 NOTES</p> <ul style="list-style-type: none"> a) TO-DO-LIST b) Command Line Interface Library c) Graphical Text User Interface Library <p>7 APPENDIXES</p> <ul style="list-style-type: none"> a) Appendix A - Nature of System and Software <p>8 TECHNICAL IMPACT ANALYSIS</p> <p>9 TEST REQUIREMENTS AND RESTRICTIONS</p> <p>10 USE CASES</p> <p>11 TRACEABILITY INFORMATION</p> <p>12 CLASS LIST BY CATEGORY</p>
8	Release Notes	<p>This document is distributed with software products, often when the product is still in the development or test state (e.g., a beta release). For products that have already been in use by clients, the release note is a supplementary document that is delivered to the customer when a bug</p>

		<p>is fixed or an enhancement is made to the product.</p> <p>Included are the following topics:</p> <ol style="list-style-type: none"> 1 SCOPE (System Specification) 2 DEFINITIONS, ABBREVIATIONS, AND ACRONYMS 3 SYSTEM REQUIREMENTS <ol style="list-style-type: none"> a) Platform Configurations (Release Notes) b) Network Configurations (Release Notes) 4 PACKAGE CONTENTS <ol style="list-style-type: none"> a) Command Line Interface Library b) Graphical Text User Interface Library 5 FIXED BUGS & ISSUES 6 KNOWN BUGS & ISSUES 7 NEW FEATURES 8 APPLICATION NOTES <ol style="list-style-type: none"> a) Installation Procedure b) User Interface Design c) Developer 9 PERFORMANCE TUNING 10 ACCEPTANCE TESTING 11 COMPONENT STATUS 12 APPENDIXES <ol style="list-style-type: none"> a) Appendix A - Baseline Host Computer Platforms b) Appendix B - API Relationship c) Appendix C - Deliverables
9	Software Users Manual	<p>This document tells a hands-on software user (developer and operator) how to install and use the associated computer software (<i>TeamSTARS</i> "tsWxGTUI_PyVx" Toolkit). It may also cover a particular aspect of software operation, such as instructions for a particular position or task.</p> <p>Included are the following topics:</p> <ol style="list-style-type: none"> 1 SCOPE (System Specification) 2 SOFTWARE SUMMARY <ol style="list-style-type: none"> a) Software Application b) Software Inventory c) Software Environment

		<ul style="list-style-type: none"> d) Software Organization and Overview of Operation e) Security and Privacy f) Assistance and Problem Reporting <p>3 ACCESS TO THE SOFTWARE</p> <ul style="list-style-type: none"> a) First-time User of the Software b) Initiating a Session c) Stopping and Suspending Work <p>4 PROCESSING REFERENCE GUIDE</p> <ul style="list-style-type: none"> a) Capabilities b) Conventions c) Processing Procedures d) Related Processing e) Data Backup f) Recovery from Errors, Malfunctions, and Emergencies g) Messages h) Quick Reference Guide <p>5 NOTES</p> <ul style="list-style-type: none"> a) Use Case(s) b) Baseline Toolkit Development Platforms <p>6 DEFINITIONS, ABBREVIATIONS, AND ACRONYMS</p> <p>7 APPENDIXES</p> <p>8 APPENDIX A - BASELINE HOST COMPUTER PLATFORMS</p> <p>9 APPENDIX B - API RELATIONSHIP</p> <p>10 APPENDIX C - DELIVERABLES</p> <p>11 APPENDIX D - LOG FILES</p>
10	Appendixes	<p>1 Introduction</p> <ul style="list-style-type: none"> a) Appendix A - Baseline Host Computer Platforms b) Appendix B - API Relationship c) Appendix C - Deliverables d) Appendix D - Accomplishments (Draft) e) Appendix E - Inherited, Field-Proven Computer Technology f) Appendix F - History of System Development. Operation, and Maintenance

		2 Software Development Plan 3 System Specification <ul style="list-style-type: none"> a) APPENDIX A - REQUIRED STATES AND MODES b) APPENDIX B - SYSTEM CAPABILITY REQUIREMENTS c) APPENDIX C - SYSTEM EXTERNAL INTERFACE REQUIREMENTS d) APPENDIX D - SYSTEM INTERNAL INTERFACE REQUIREMENTS e) APPENDIX E - SYSTEM INTERNAL DATA REQUIREMENTS f) APPENDIX F - ADAPTATION REQUIREMENTS g) APPENDIX G - SAFETY REQUIREMENTS h) APPENDIX H - SECURITY AND PRIVACY REQUIREMENTS i) APPENDIX I - SYSTEM ENVIRONMENT REQUIREMENTS j) APPENDIX J - COMPUTER RESOURCE REQUIREMENTS k) APPENDIX K - SYSTEM QUALITY FACTORS l) APPENDIX L - DESIGN AND CONSTRUCTION CONSTRAINTS m) APPENDIX M - PERSONNEL-RELATED REQUIREMENTS n) APPENDIX N - TRAINING-RELATED REQUIREMENTS o) APPENDIX O - LOGISTICS-RELATED REQUIREMENTS p) APPENDIX P - OTHER REQUIREMENTS q) APPENDIX Q - PACKAGING REQUIREMENTS r) APPENDIX R - PRECEDENCE AND CRITICALITY OF REQUIREMENTS 4 Interface Requirements Specification 5 Software Requirements Specification <ul style="list-style-type: none"> a) APPENDIX A - Nature of System and Software 6 Release Notes <ul style="list-style-type: none"> a) APPENDIX A - Baseline Host Computer Platforms
--	--	--

		<p>b) APPENDIX B - API Relationship</p> <p>c) APPENDIX C - Deliverables</p> <p>7 Software Users Manual</p>
11	Dictionary	A reference document that contains words, phrases, abbreviations and acronyms that are listed in alphabetical order with information about the their meanings in the context of the <i>TeamSTARS</i> "tsWxGTUI_PyVx" Toolkit software.
12	To-Do	

Draft

2 SOFTWARE SUMMARY

This section shall be divided into the following paragraphs.

- *Software Application* (on page 21)
- *Software Inventory* (on page 37)
- *Software Environment* (on page 39)
- *Software Organization and Overview of Operation* (on page 44)
- *Security and Privacy* (on page 78)
- *Assistance and Problem Reporting* (on page 79)

2.1 Software Application

This paragraph shall provide a brief description of the intended uses of the software. Capabilities, operating improvements, and benefits expected from its use shall be described.

- *Capabilities* (on page 21)
- *Operating Improvements* (on page 36)
- *Benefits* (on page 36)

2.1.1 Capabilities

This paragraph shall provide a brief description of the capabilities, operating improvements, and benefits expected from the "tsWxGTUI" Toolkit.

- *Requirements and Constraints (Development Plan)* (on page 22)
- *Operating Improvements* (on page 36)
- *Benefits* (on page 36)

2.1.1.1 Requirements and Constraints (Development Plan)

This paragraph shall provide a brief description of the "tsWxGTUI_PyVx" Toolkit's features, capabilities and limitations.

- **Goals & Capabilities (Development Plan)** (on page 22) - Describes the Functional & Interface Requirements
- **Non-Goals & Limitations (Development Plan)** (on page 35) - Describes the Design Constraints
- **Comparison of wxPython with tsWxGTUI (Development Plan)** (on page 159) - Compares the features, capabilities and limitations of the pixel-mode "wxPython" / "wxWidgets" / "wxEmbedded" Toolkit with those of its character-mode emulator, the "tsWxGTUI_PyVx" Toolkit.

2.1.1.1.1 Goals & Capabilities (Development Plan)

This section summarizes the requirements implicit and explicit in the "tsWxGTUI_PyVx" Toolkit's purpose. As the need arises for their specific skills (See **DEFINITIONS, ABBREVIATIONS, AND ACRONYMS** (on page 125)), System Administrators, Software Engineers, System Operators and Field Service Personnel become the operator of the computer system.

This is a high-level view of functional, interface, design, implementation, operation, quality and reliability requirements that are within the work scope.

- 1 A means for one or more operators to use local and remote computer operating systems, as appropriate to the application:
 - a) Concurrently and/or sequentially login to local and remote computer operating systems.
 - b) Concurrently and/or sequentially launch, interact with and terminate one or more operating system and/or application processes which may concurrently and/or sequentially launch, interact with and terminate one or more associated task threads.
 - c) Concurrently and/or sequentially logoff local and remote computer operating systems.
- 2 A means for one or more operators to interactively monitor and control local and remote computer equipment via either or both of the following, as appropriate to the application:
 - a) Command Line Interface (CLI) --- Must support the Python 2.6.8 and Python 3.2.3 releases and compatible later and earlier releases.
 - b) Graphical User Interface (GUI) --- Must support the character-mode compatible emulation of the Application Programming Interface of the wxPython 2.8.9.2 release and compatible later and earlier releases.
- 3 A means for the application developer to interact with the local and remote computer equipment and operator(s) via:
 - a) Popular, cross-platform Application Programming Interfaces (APIs). The Command Line Interface and Graphical User Interface APIs must be free, open source and field-proven. The APIs must also have an ongoing and extensive track record of active use, maintenance and enhancement.
 - b) Libraries of building block modules, tests, tools and utilities

- c) Designs suitable for installation and use in embedded system which typically have limited, application-specific processing, memory, communication, input/output and file storage resources. Some systems may only have character-mode operator interface hardware suitable for the host computer operating system's command line interface console.
 - d) CLI and GUI interfaces that are attractive and user friendly.
- 4 A means for the developer to re-use an existing desktop, laptop, workstation and embedded system applications on an extensive variety of platforms including cell phones, laptops, desktops, workstations and super computers with 32-bit and 64-bit processors from various manufacturers.
 - 5 A means for the developer to re-use built-in package and module documentation (see ***Built-in Documentation Symbols*** (on page 24)) to facilitate development of the Command Line Interface and Graphical-style User Interface.
 - 6 A means for the developer to launch application programs (see ***Application Launch Modules*** (on page 29)) to facilitate development of the Command Line Interface and Graphical-style User Interface.
 - 7 A means for the developer to import packages and modules (see ***Directory and Import Guidelines*** (on page 32)) to facilitate development of the Command Line Interface and Graphical-style User Interface.
 - 8 A means for the developer to install and troubleshoot modified packages and modules (see ***Site Package Installation Guidelines*** (on page 35)) to facilitate development of the Command Line Interface and Graphical-style User Interface.
 - 9 A means for the developer to troubleshoot design and user issues via date and time stamped configuration, operational, diagnostic and exception logs.
 - 10 A means for the developer to co-ordinate the automated operation and failure recovery of multiple applications by use of Unix-style exit codes and error messages.
 - 11 A means for the operator to use a mouse, trackball or touchpad to select and manipulate GUI objects when the platform supports such a device. Support shall include:
 - a) Left, Right and Wheel/Middle mouse button features such as single-click, double-click and triple-click.
 - b) Scrollbar features such as Left, Right, Up and Down arrow.
 - c) Scrollbar and Slider gauge display whose horizontal or vertical bar shows the relative position (via a single non-blank fill character) and size (via additional non-blank fill characters) of the view area within the associated scrolled text. Gauges are also clickable windows that trigger an associated event (such as scroll the associated displayed text left, right, up or down). A single Left-Click on a ScrollBarGauge moves the text toward or away from the top (first) or bottom (last) horizontal or vertical character in proportion to how close the mouse caret (pointer) was to the arrow buttons at either end of the associated gauge. The relative position and size of the highlighted area in the ScrollBarGauge alerts the operator to the availability of text currently hidden from view. It eliminates any justification for wasting screen real estate by alerting the operator to hidden text via the tilde ("~") character.
 - 12 A means for the operator to use a keyboard to select GUI objects when the platform does not support a mouse, trackball or touchpad.
 - 13 A means for the operator to use a Task bar (a top-level window) that contains buttons for each application frame and dialog window that can be used to control which application frame or dialog window has focus and is not partially hidden behind any other frame and dialog window.

- 14** A means (Site-Packages) for the application and toolkit developer to import application and library building-block components from a single-level (Global Module Index-style) directory file system. This is intended to extend the capabilities of the standard Python library packages and modules by the standard procedures for installing and using those third-party library packages and modules which are not released by the Python Software Foundation.
- 15** A means (Developer-Sandboxes) for the application and toolkit developer to import application and library building-block components from a nested, multilevel directory file system. This is intended to facilitate development without and before the creation and installation of released Site-Packages. This eliminates the need for building and installing candidate bug fixes to the Site-Packages before they can even be tested.
- 16** Sufficient system hardware and software reliability, availability and maintainability to continuously operate unattended and without failure for extended periods of time on an "AS IS" basis unless otherwise certified by suitably qualified system integrators.

2.1.1.1.1 Built-in Documentation Symbols

The Command Line Interface module "tsApplication.py" in the "tsLibCLI" library defines and distributes a set of private symbols (those having "__" as both a prefix and suffix) that are passed, via public symbols, and thereby shared with other modules to avoid duplication of data and extra developer effort.

During application program launch:

- The Command Line Interface (CLI) module "tsCommandLineEnv.py" in the "tsLibCLI" library passes the set of CLI public symbols to either the default "tsOperatorSettingsParser" module in the "tsLibCLI" library or to an application-specific surrogate.
- The Graphical User Interface (GUI) module "tsWxMultiFrameEnv.py" in the "tsLibGUI" library passes the set of CLI public symbols and the set of GUI public symbols to either the default "tsOperatorSettingsParser" module in the "tsLibCLI" library or to an application-specific surrogate.

PUBLIC SYMBOL	PRIVATE SYMBOL	DEFINITION	NOTES
			<p>(1) Future Python releases may enforce adoption of a strict adherence to "PEP 8 - Style Guide for Python Code".</p> <p>(2) Justification for violation of "PEP 8":</p> <ul style="list-style-type: none"> ▪ Minimizes the probability that software developers will alter or override and thereby interfere with the use of these private symbols. ▪ Isolated sharing of private symbols, via their public names, occurs only during launcher invocation. ▪ During unit testing of any Python module, the "__header__" private symbol will be available for display by the software developer or tester. ▪ During the launch of any Python application program with the "-h" or "--help" command line option, the program's "mainTitleVersionDate" and "__doc__"

			private symbol will be available for display by the software developer, tester or user.
buildPurpose	__doc__	Custom formatted Python docstring describing Package or Module Purpose	<p>(1) For an example see "./tsDemoArchive/tsTestsLibCLI/test_tsCommandLineEnv.py"</p> <p>(2) Python will implicitly use the identifier ("__doc__") for docstrings.</p> <p>(3) The "tsStripComments.py" module, in the "tsToolsCLI" utility package, will NOT remove those docstrings that are associated with an assignment statement (identifier = """ text string(s) """ or identifier = " text string(s) "). The identifier may be explicitly named "__doc__".</p> <p>(4) The "tsStripComments.py" module, in the "tsToolsCLI" utility package, will remove those docstrings (""" text string(s) """ or " text string(s) ") that are NOT associated with an assignment statement. This typically results in application program traps when the "tsOperatorSettingsParser" module references non-existent information needed for built-in help.</p>
buildTitle	__title__	Custom formatted Python text string identifies Package or Module Name	(1) For an example see "./tsDemoArchive/tsTestsLibCLI/test_tsCommandLineEnv.py"
buildVersion	__version__	Custom formatted Python text string identifies Package or Module Version	(1) For an example see "./tsDemoArchive/tsTestsLibCLI/test_tsCommandLineEnv.py"
buildDate	__date__	Custom formatted Python text string identifies Package or Module modified Date	(1) For an example see "./tsDemoArchive/tsTestsLibCLI/test_tsCommandLineEnv.py"
buildAuthors	__authors__	Custom formatted Python text string identifies Package or Module Author(s)	(1) For an example see "./tsDemoArchive/tsTestsLibCLI/test_tsCommandLineEnv.py"
buildCopyright	__copyright__	Custom formatted Python text string identifies Package or Module Copyright notice	(1) For an example see "./tsDemoArchive/tsTestsLibCLI/test_tsCommandLineEnv.py"
buildLicense	__license__	Custom formatted Python text string identifies Package	(1) For an example see "./tsDemoArchive/tsTestsLibCLI/test_tsCommandLineEnv.py"

		or Module License notice	
buildCredits	__credits__	Custom formatted Python text string identifies Package or Module Third-party Author(s)	(1) For an example see <code>"/tsDemoArchive/tsTestsLibCLI/test_tsCommandLineEnv.py"</code>
	__line1__	Custom formatted Python text string formats Package or Module Title, Version and Modified Date header line(s) for display at run time	(1) For an example see <code>"/tsDemoArchive/tsTestsLibCLI/test_tsCommandLineEnv.py"</code>
	__line2__	Custom formatted Python text string formats Package or Module Author(s) header line(s) for display at run time	(1) For an example see <code>"/tsDemoArchive/tsTestsLibCLI/test_tsCommandLineEnv.py"</code>
	__line3__	Custom formatted Python text string formats Package or Module Copyright header line(s) for display at run time	(1) For an example see <code>"/tsDemoArchive/tsTestsLibCLI/test_tsCommandLineEnv.py"</code>
	__line4__	Custom formatted Python text string formats Package or Module License and Third-party Credits header line(s) for display at run time	(1) For an example see <code>"/tsDemoArchive/tsTestsLibCLI/test_tsCommandLineEnv.py"</code>
buildHeader	__header__	Custom formatted Python text string formats Package or Module header line(s) 1, 2, 3 and 4 for display at run time	(1) For an example see <code>"/tsDemoArchive/tsTestsLibCLI/test_tsCommandLineEnv.py"</code>
buildTitleVersionDate	mainTitleVersionDate	Custom formatted Python text string formats Package or Module Title, Version, modification Date header line for display at run time	(1) For an example see <code>"/tsDemoArchive/tsTestsLibCLI/test_tsCommandLineEnv.py"</code>
enableDefaultComma	Not Applicable	Python boolean	(1) For an example see

ndLineParser		value of "True" or "False".	<p>"/tsDemoArchive/tsTestsLibCLI/test_tsCommandLineEnv.py"</p> <p>(2) Use "True" to enable use of the one and only operator settings parser in "tsLibCLI"</p> <p>(3) Use "False" to disable use of the one and only operator settings parser in "tsLibCLI" and the application must supply a parser that handles -h/--help, -a/--about, -v/--version, -d/--debug, -V/--Verbose and any other keyword-value pair options and/or positional arguments.</p>
guiMessageFilename	Not Applicable	Python Redirected Output file path and name.	<p>(1) For an example see "/tsDemoArchive/tsTestsLibGUI/test_tsWxMultiFrameEnv.py"</p> <p>(2) Use Python's None to designate default path and name</p> <p>(3) Use a Python text string to designate an application-specific path and name.</p>
guiMessageRedirect	Not Applicable	Python boolean value of "True" or "False".	<p>(1) For an example see "/tsDemoArchive/tsTestsLibGUI/test_tsWxMultiFrameEnv.py"</p> <p>(2) Use "True" to enable redirected output.</p> <p>(3) Use "False" to disable redirected output.</p>
guiRequired	Not Applicable	Python boolean value of "True" or "False".	<p>(1) For an example see "/tsDemoArchive/tsTestsLibGUI/test_tsWxMultiFrameEnv.py"</p> <p>(2) Use "True" to use GUI mode of operation.</p> <p>(3) Use "False" to use CLI mode of operation</p>
guiTopLevelObject	Not Applicable	Python instance of top-level wxPython-style GUI Frame object.	<p>(1) For an example see "/tsDemoArchive/tsTestsLibGUI/test_tsWxMultiFrameEnv.py"</p>
guiTopLevelObjectId	Not Applicable	Python integer value for top-level wxPython-style GUI Frame object ID	<p>(1) For an example see "/tsDemoArchive/tsTestsLibGUI/test_tsWxMultiFrameEnv.py"</p> <p>(2) Assign a specific value or use wx.ID_ANY to get the default, automatically assigned value</p>
guiTopLevelObjectName	Not Applicable	Python text string value for top-level wxPython-style GUI Frame object Name	<p>(1) For an example see "/tsDemoArchive/tsTestsLibGUI/test_tsWxMultiFrameEnv.py"</p>
guiTopLevelObjectParent	Not Applicable	Python instance of top-level wxPython-style GUI Frame object's parent or None as appropriate.	<p>(1) For an example see "/tsDemoArchive/tsTestsLibGUI/test_tsWxMultiFrameEnv.py"</p>

guiTopLevelObjectPosition	Not Applicable	Python instance of top-level wxPython-style GUI Frame object's position or wx.DefaultPosition as appropriate.	(1) For an example see <code>"/tsDemoArchive/tsTestsLibGUI/test_tsWxMultiFrameEnv.py"</code>
guiTopLevelObjectSize	Not Applicable	Python instance of top-level wxPython-style GUI Frame object's size or wx.DefaultSize as appropriate.	(1) For an example see <code>"/tsDemoArchive/tsTestsLibGUI/test_tsWxMultiFrameEnv.py"</code>
guiTopLevelObjectStatusBar	Not Applicable	Python instance of top-level wxPython-style GUI Frame object's StatusBar or None as appropriate.	(1) For an example see <code>"/tsDemoArchive/tsTestsLibGUI/test_tsWxMultiFrameEnv.py"</code>
guiTopLevelObjectStyle	Not Applicable	Python instance of top-level wxPython-style GUI Frame object's Style or wx.DefaultStyle as appropriate.	(1) For an example see <code>"/tsDemoArchive/tsTestsLibGUI/test_tsWxMultiFrameEnv.py"</code>
guiTopLevelObjectTitle	Not Applicable	Python instance of top-level wxPython-style GUI Frame object's Title or wx.DefaultTitle as appropriate.	(1) For an example see <code>"/tsDemoArchive/tsTestsLibGUI/test_tsWxMultiFrameEnv.py"</code>
logs	Not Applicable	Python list of text strings for any named log file or an empty list as appropriate.	(1) For the basic CLI example see <code>"/tsDemoArchive/tsTestsLibCLI/test_tsApplication.py"</code> (2) For the CLI Environment Wrapper example see <code>"/tsDemoArchive/tsTestsLibCLI/test_tsCommandLineEnv.py"</code> (3) For the GUI Environment Wrapper example see <code>"/tsDemoArchive/tsTestsLibGUI/test_tsWxMultiFrameEnv.py"</code>
runTimeEntryPoint	Not Applicable	Python instance of the entry point for either the application program or its environment wrapper as appropriate.	(1) For the basic CLI example see <code>"/tsDemoArchive/tsTestsLibCLI/test_tsApplication.py"</code> (2) For the CLI Environment Wrapper example see <code>"/tsDemoArchive/tsTestsLibCLI/test_tsCommandLineEnv.py"</code> (3) For the GUI Environment Wrapper example see

			"/tsDemoArchive/tsTestsLibGUI/test_tsWxMultiFrameEnv.py"

2.1.1.1.2 Application Launch Modules

The following Application Launch Building-block Modules facilitate development of the Command Line Interface and Graphical-style User Interface.

- 1 **"tsApplication"** - Base class to initialize and configure the application program launched by an operator. It delivers those keyword-value pair options and positional arguments specified by the operator, on the command line, and by the application, in its invocation parameter list. It initializes any application specific logger(s). It enables an application launched via a Command Line Interface (CLI) to initialize, configure and use the same character-mode terminal with a Graphical-style User Interface (GUI).
- 2 **"tsCommandLineEnv"** - Class, that uses **"tsApplication"**, to initialize and configure the application program launched by an operator. It delivers those keyword-value pair options and positional arguments specified by the operator, on the command line, and by the application, in its invocation parameter list. It initializes any application specific logger(s). It wraps the Command Line Interface application with exception handlers to control exit codes and messages that may be used to coordinate other application programs.
- 3 **"tsWxMultiFrameEnv"** - Class, that uses **"tsCommandLineEnv"**, to enable an application using a Command Line Interface (CLI) to launch and use the same character-mode terminal with a Graphical-style User Interface (GUI). It delivers those keyword-value pair options and positional arguments specified by the operator, on the command line, and by the application, in its invocation parameter list. It initializes any application specific logger(s). It wraps the CLI, underlying the GUI, and the GUI with exception handlers to control the exit codes and messages used to coordinate other application programs.

Upon Application program launch, the "tsCommandLineEnv" or "tsWxMultiFrameEnv" modules invoke the "tsApplication" module to pass the **Built-in Documentation Symbols** (on page 24) to the default or application-specific "tsOperatorSettingsParser" as described below.

```
# Usage (example):
#
#     ## Add the following source code to the main application
#     ## program file.
#     ##
#     ## Note: The application is responsible for collecting,
#     ##         parsing and applying command line keyword-value
#     ##         pair options and positional arguments. The
#     ##         application is also responsible for wrapping
#     ##         itself with application-specific exception
#     ##         handlers that set exit codes and messages,
#     ##         upon termination, that may be used to coordinate
#     ##         the actions of other applications.
#
#     ## Import Module
#     import tsApplication as tsAPP
#
#     ## Generalized Form to Instantiate and Launch an application
#     ## module that uses a Command Line Interface (CLI) to a
#     ## character-mode terminal with optional logging.
#     ##
#     ## Configuration options enable the CLI application to launch
#     ## and use the same character-mode terminal with a Graphical-
#     ## style User Interface (GUI).
#     ##
#     ## See this File's header for examples of those application
#     ## specific source code descriptions associated with
#     ## parameter identifiers having double-underscore ("__")
#     ## prefix and suffix.
#     ##
#     ## See the test_tsWxWidgets.py File's header for examples of
#     ## the gui application options.
#
# -----
#
#     #####
#
#     myApp = tsAPP.TsApplication(
#
#         #####
#         # All applications (with Command Line Interface or
#         # Graphical-style User Interface) begin with the following
#         # Command Line Interface Launch configuration item list:
#
#         buildTitle=__title__,
#         buildVersion=__version__,
#         buildDate=__date__,
#         buildAuthors=__authors__,
#         buildCopyright=__copyright__,
#         buildLicense=__license__,
#         buildCredits=__credits__,
#         buildTitleVersionDate=mainTitleVersionDate,
#         buildHeader=__header__,
#         buildPurpose=__doc__,
#
#         #####
```



```

#
# # Python version appropriate Command Line Interface
# # module(s) may be enabled to obtain non-Application-
# # specific Keyword-Value pair Options and Positional
# # Arguments and associated command line help:
#
# # "argparse" module - introduced with Python 2.7.0
# # "optparse" module - introduced with Python 2.3.0
# # "getopt" module - introduced with Python 1.6.0
#
enableDefaultCommandLineParser=False # Disable unless True
#
#####
#
# When appropriate, some applications also use the following
# Graphical-style User Interface Launch configuration item list:
#
guiMessageFilename=None,
guiMessageRedirect=True,
guiRequired=True,
guiTopLevelObject=_Communicate,
guiTopLevelObjectId=wx.ID_ANY,
guiTopLevelObjectName='Sample',
guiTopLevelObjectParent=None,
guiTopLevelObjectPosition=wx.DefaultPosition,
guiTopLevelObjectSize=wx.DefaultSize,
guiTopLevelObjectStatusBar=None,
guiTopLevelObjectStyle=wx.DEFAULT_FRAME_STYLE,
guiTopLevelObjectTitle='widgets_communicate',
#
#####
#
# When customized logging is appropriate, some applica-
# tions use the following application-specific Launch
# configuration item:
#
logs=['1st-Non-Default', ..., 'Nth-Non-Default'],
#
# When basic logging is appropriate, some applications
# use the following non-application-specific Launch
# configuration item:
#
logs=[],
#
#####
#
# All applications, with Command Line Interface or with
# both Command Line and Graphical-style User Interfaces,
# wrapup their Configuration item list as follows:
#
runTimeEntryPoint=main)
#
#####
#
#####
#

```

```
#      ## Simplest Form to Instantiate Module with only standard logging
#      myApp = tsAPP.TsApplication(
#          buildTitle=__title__,
#          buildVersion=__version__,
#          buildDate=__date__,
#          buildTitleVersionDate=mainTitleVersionDate,
#          buildHeader=__header__,
#          buildPurpose=__doc__,
#          logs=[],
#          runTimeEntryPoint=main)
#
#      #####
#
#      ## Simplest Form to Instantiate Module with custom logging
#      myApp = tsAPP.TsApplication(
#          buildTitle=__title__,
#          buildVersion=__version__,
#          buildDate=__date__,
#          buildTitleVersionDate=mainTitleVersionDate,
#          buildHeader=__header__,
#          logs=['1st-Non-Default', '2nd-Non-Default', '3rd-Non-Default'],
#          runTimeEntryPoint=main)
#
#      #####
#
#      ## Launch via reference to appropriate Module Method
#      myApp.runMainApplication()
```

2.1.1.1.1.3 Directory and Import Guidelines

The "tsWxGTUI" Toolkit uses a non-traditional mechanism to facilitate the debugging and non-duplicating import of building block library packages. The mechanism uses the "__init__.py" module in each package to establish the package's component modules and the package's relationship to its ancestors in the "tsWxGTUI" Toolkit package hierarchy.

Representative applications:

- 1 CLI mode: tsToolsCLI/tsPlatformQueryPkg/src/tsPlatformQuery.py
- 2 GUI mode: tsLibGUI/tsWxPkg/test/test_tsWxMultiFrameEnv.py

Draft

The following code fragment illustrates an explicit way to handle the package and module dependency of the GUI features on the CLI ones:

```
__title__ = 'YourApplicationName'

GUImode = False
CLImode = not GUImode

# Python-based CLI Mode
try:

    import tsLibCLI

    import tsExceptions as tse
    import tsLogger as Logger
    import tsOperatorSettingsParser

    from tsDoubleLinkedList \
        import DoubleLinkedList

    if CLImode:

        from tsCommandLineEnv \
            import CommandLineEnv

except ImportError, importCode:

    fmt1 = '%s: ImportError ' + \
        '(tsLibCLI); ' % __title__
    fmt2 = 'importCode=%s' % str(importCode)
    msg = fmt1 + fmt2

    print(msg)

    raise tse.PROGRAM_EXCEPTION(
        tse.APPLICATION_TRAP,
        msg)

if GUImode:

    # wxPython-style, nCurses-based GUI Mode
    try:

        import tsLibGUI

        import tsWx as wx
        from tsWxMultiFrameEnv import MultiFrameEnv

    except ImportError, importCode:

        fmt1 = '%s: ImportError ' + \
            '(tsLibGUI); ' % __title__
        fmt2 = 'importCode=%s' % str(importCode)
        msg = fmt1 + fmt2
```

```

print(msg)

raise tse.UserInterfaceException(
    tse.CHARACTER_GRAPHICS_NOT_AVAILABLE,
    msg)

```

2.1.1.1.4 Site Package Installation Guidelines

It is recommended that the "tsWxGTUI" Toolkit evaluators download and unpack the compressed "tarball" or "zip" file but NOT use "pip" or "setup.py" to install it with other Python version specific site packages.

If you install the site package, you will not be able to use or debug those components which you have modified but not yet installed.

2.1.1.1.2 Non-Goals & Limitations (Development Plan)

This section summarizes those requirements explicitly beyond the scope of the "tsWxGTUI_PyVx" Toolkit's purpose.

This is a high-level view of functional, interface, design, implementation, operation, quality and reliability requirements that are beyond the work scope.

- 1 A means to emulate each and every capability of the Application Programming Interface (API) of the pixel-mode Graphical User Interface (such as the C++ language based "wxWidgets" and its Python language based "wxPython" wrapper) because the "tsWxGTUI_PyVx" Toolkit is designed for embedded systems which typically have limited, application-specific processing, memory, communication, input/output and file storage resources. Some may only have character-mode hardware suitable for their operating system's command line console.
- 2 A means to develop support for pixel-mode GUI features such as icons, proportional fonts and other bit-mapped images because many embedded system displays operate only in character-mode.
- 3 A means to programmatically re-size the top level application Frame or any other GUI object because:
 - a) The host operating system's command line shell window is opened upon operator login, or subsequent launch command, and the size of the shell window establishes the initial size of the character-mode display (such as "stdscr", the Python "Curses" or GNU "nCurses" low-level GUI-style standard screen). Once a launched CLI or GUI-style application has been terminated, the System Operator may change the size of the shell window. The position of the shell window may be changed at any time.
 - b) The application that directly use the character-mode display (identified as "stdscr" in Python "Curses" or GNU "nCurses") are efficient enough to respond to operator re-sizing events related to the top-level "stdscr" or any associated GUI -style objects. They can also use the "pad" feature to create virtual windows whose out-of-bounds contents are clipped (not displayed) without triggering an error. Those application are responsible for programmatically re-sizing low-level GUI objects and any dependent ones.
 - c) It is not yet known how to make applications programmatically re-size any or all of the high- and low-level GUI-style objects and any dependent ones, without a complete, time consuming application restart.

- 4 A means and expertise to re-compile and re-build the "nCurses" library and associated Python wrapper with the wide character option needed to support Unicode characters and 256-colors because its installation could adversely effect the operation of host operating system utilities.
- 5 A means and expertise to extend the Python Curses module to include support for all of the currently available "nCurses" methods and functions.

The use of Python C-language extensions is beyond the work scope because the cross-platform "tsWxGTUI_PyVx" Toolkit is intended to be used on host computer platforms for which the Toolkit developer does not have the access required for qualification testing.

From "Extending Python with C or C++":

"It is quite easy to add new built-in modules to Python, if you know how to program in C. Such extension modules can do two things that can't be done directly in Python: they can implement new built-in object types, and they can call C library functions and system calls.

To support extensions, the Python API (Application Programmers Interface) defines a set of functions, macros and variables that provide access to most aspects of the Python run-time system. The Python API is incorporated in a C source file by including the header "Python.h".

The compilation of an extension module depends on its intended use as well as on your system setup; details are given in later chapters.

Do note that if your use case is calling C library functions or system calls, you should consider using the ctypes module rather than writing custom C code. Not only does ctypes let you write Python code to interface with C code, but it is more portable between implementations of Python than writing and compiling an extension module which typically ties you to CPython."

- 6 A means and expertise to resolve anomalies in the look and feel of vt100, vt220, xterm, xterm-color, xterm-16color, xterm-88color and xterm-256color terminal emulations that are platform-specific.

2.1.2 Operating Improvements

This paragraph shall provide a brief description of the "tsWxGTUI" Toolkit's operating improvements.

- **Comparison of wxPython with tsWxGTUI** (see "**Comparison of wxPython with tsWxGTUI (Development Plan)**" on page 159) - Tabulation of features, capabilities and limitations.
- **Appendix B - API Relationship** (on page 159) - Tabulation of features, capabilities and limitations of the High-level, Low-Level and Extended APIs associated with the "tsWxGTUI" Toolkit and its third-party components.

2.1.3 Benefits

The toolkit facilitates the creation of "user friendly" applications.

- 1 **Command Line Interface (tsToolkitCLI)** - Includes building blocks that create a sophisticated UNIX-like terminal interface featuring: exception handling, logging and the launching, event dispatching and terminating of the Graphical-style User Interface.
- 2 **Graphical-style User Interface (tsToolkitGUI)** - Includes building blocks that create a sophisticated Desktop, Laptop and Workstation Computer-like terminal interface featuring:

- a) tiled and overlapped windows
- b) box and grid sizers
- c) buttons
- d) check boxes
- e) horizontal and vertical gauges
- f) menu bars and menus
- g) radio boxes and buttons
- h) scrolled windows with associated scrollable text window and scrollbars with associated gauge and scroll control buttons
- i) redirected output window to annotate date, time and severity levels to system and application messages printed or sent to stderr and stdout
- j) splash screen
- k) multi-field status bars
- l) tool bars
- m) task bar

3 Improved Productivity - The general purpose, re-usable building blocks enable the application developer to focus on the application specific functionality and not waste effort re-inventing the functionality typical of Command Line and Graphical User Interfaces. It enables "wxPython" pixel-mode applications to run with little, if any, change if they neither use icons or other bit-mapped images, nor use proportional sized fonts or associated special features.

2.2 Software Inventory

This paragraph shall identify all software files, including databases and data files, that must be installed for the software to operate. The identification shall include security and privacy considerations for each file and identification of the software necessary to continue or resume operation in case of an emergency.

The "tsWxGTUI" Toolkit and its third-party components are developed and distributed in a manner typical of other Python programs.

Individual releases are compatible only with one or more designated Python versions. For example, a release for Python 2.7.x fixes bugs but otherwise maintains Application Programming Interface backward compatibility with 2.7.0. It may also maintain backward compatibility to Python 2.6.0. However, it may not maintain backward compatibility to Python 2.5.0 or forward compatibility to Python 3.0.

- **Manifest for Python-2.x** (on page 38) - Compatibility for Python 2.6.4-2.7.6 spot-checked on Linux, Mac OS X, Microsoft Windows and Unix
- **Manifest for Python-3.x** (on page 39) - Compatibility for Python 3.1.0-3.2.3 spot-checked on Linux, Mac OS X and Microsoft Windows

2.2.1 Manifest for Python-2.x

See the "MANIFEST" file that accompanied the installed "tsWxGTUI" Toolkit release. That release is intended only for use with the legacy and present Python 2.6-2.7 versions. Modifications may be required to backport a copy of the Toolkit for use with Python 2.0-2.5.

It identifies all software files, including databases and data files, that must be installed for the software to operate. The identification includes security and privacy considerations for each file and identification of the software necessary to continue or resume operation in case of an emergency.

The "**MANIFEST_TREE.sh**" shell script has also been provided to generate a file and directory listing (annotated with access permissions and the date and time of the last modification) in both plain text and HTML formats.

NOTES:

1. The "tsWxGTUI" Toolkit and its third-party components are distributed as free and open source software. You may use, modify and redistribute individual copyrighted works only under the terms and conditions of the license designated by the original work's author(s). See the accompanying "COPYRIGHT.txt" and "LICENSE.txt" files for details.
 2. In Unix-like operating systems, "chmod" is the name of a shell command and a system call, which both change the access permissions to file system objects (including files and directories). The name is an abbreviation of change mode. The default mode (in octal format for user, group and others) is "755".
-

2.2.2 Manifest for Python-3.x

See the "MANIFEST" file that accompanied the installed "tsWxGTUI" Toolkit release. That release is intended only for use with the Python 3.2 versions. Modifications may be required to backport a copy of the Toolkit for use with present and future Python 3.0-3.1 and Python 3.3-3.4 versions.

It identifies all software files, including databases and data files, that must be installed for the software to operate. The identification includes security and privacy considerations for each file and identification of the software necessary to continue or resume operation in case of an emergency.

The "**MANIFEST_TREE.sh**" shell script has also been provided to generate a file and directory listing (annotated with access permissions and the date and time of the last modification) in both plain text and HTML formats.

NOTES:

1. The "tsWxGTUI" Toolkit and its third-party components are distributed as free and open source software. You may use, modify and redistribute individual copyrighted works only under the terms and conditions of the license designated by the original work's author(s). See the accompanying "COPYRIGHT.txt" and "LICENSE.txt" files for details.
 2. In Unix-like operating systems, "chmod" is the name of a shell command and a system call, which both change the access permissions to file system objects (including files and directories). The name is an abbreviation of change mode. The default mode (in octal format for user, group and others) is "755".
-

2.3 Software Environment

This paragraph shall identify the hardware, software, manual operations, and other resources needed for a user to install and run the software.

- **Computer Equipment** (on page 40)
- **Communications Equipment** (on page 41)
- **Other Software** (on page 41)
- **Forms, Procedures, or Other Manual Operations** (on page 43)
- **Other Facilities, Equipment, or Resources** (on page 44)

2.3.1 Computer Equipment

This paragraph shall identify the computer equipment that must be present, including amount of memory needed, amount of auxiliary storage needed, and peripheral equipment such as printers and other input/output devices.

1 Local (with or without one or more optional remote) Host Computers

The computer equipment used develop the "tsWxGTUI" Toolkit and application programs should be chosen to reflect the user's goals and constraints.

- a) The platform must have a 32-bit/64-bit processor and sufficient mermory, auxiliary storage and peripheral equipment for software development and/or equipment operator use.
- b) The platform must have a POSIX-compatible operating system that supports Python software development and the execution of applications written in the Python 2.x and/or Python 3.x programming language.

2 Local (with or without one or more optional remote) Operator's Terminals

- a) Keyboard device(s)
- b) Optional Pointing device(s)
 - Mouse (or Trackball) with Left Button, Right Button and Optional Wheel/Middle Button
 - Touch Screen (Trackpad or Touchpad) using gestures to emulate mouse and buttons.
- c) Optional Printer device(s) (Impact, Ink or Laser) with output exceeding 120 characters per second.
- d) Display device(s)

multi-color terminal or terminal emulator (xterm with red-green-blue phosphors for displaying 8, 16, 88 or 256 colors)

two-color terminal or terminal emulator (vt100 or vt220 with single phosphor for displaying color-on-black or black-on-color)

Suggested platform types and display resolutions suitable for "tsxGTUI" Toolkit use. Resolution nominclature From Wikipedia, the free encyclopedia:

Use	Screen Size	Standard Monitor	High Quality Monitor	Ultra Quality Monitor
Embedded	12 inch diag.	640 x 480 VGA		
Laptop	15 inch diag.	800 x 600 SVGA	1024 x 768 XGA	1152 x 864 XGA+
Laptop	17 inch diag.	1024 x 768 XGA	1280 x 960 SXGA-	1280 x 1024 SXGA
Desktop	19 inch diag.	1280 x 960 SXGA	1280 x 1024 SXGA	1600 x 1200 UXGA
Desktop	21 inch diag.	1600 x 1200 UXGA	1920 x 1200 WUXGA	2048 x 1152 QWXGA
Workstation	27 inch diag.	1920 x 1200 WUXGA	2560 x 1440 WQHD	2560 x 1600 WQXGA

2.3.2 Communications Equipment

This paragraph shall identify the communications equipment that must be present.

1 Local (with or without one or more optional remote) Network Communications Interfaces

- a) Optional 100 Mb to 1 Gbps Ethernet Adapter and Wired or Wireless Devices suitable for Local Area Network and/or Wide Area Network interconnect circuits.
- b) Optional 100 Mb to 1 Gbps WIFI Adapter and Wired or Wireless Devices suitable for Local Area Network and/or Wide Area Network interconnect circuits.
- c) Optional 1200 bps to 33.6 Kbps Modem and Wired or Wireless Devices suitable for Local Area Network and/or Wide Area Network interconnect circuits.

2.3.3 Other Software

This paragraph shall identify the other software that must be present, such as operating systems, databases, data files, utilities, and other supporting systems.

- 1 **Host Computer** - Operating system (Linux, Mac OS X, Microsoft Windows [with free Cygwin POSIX-compliant Command Line Interface and GNU toolkit add-on from Red Hat], Unix), device drivers, run time libraries, file system manager, terminal emulators (X11, xterm, xterm-color, xterm-16color, xterm-88color, xterm-256color and vt100/vt220), command line shell (bash)
- 2 **Software Development Tools** - Python (2.6.x-2.7.x or 3.0.x-3.4.x with their associated Virtual Machine and Run Time Libraries), text or source code editors (xemacs), optional file and folder comparison and synchronization software application/diff tool (DeltaWalker, Kdiff3), standard (Python IDLE) or optional Python program debugger (Wing IDE Pro 4 and 5)
- 3 **High-Level Terminal Interface Library** - The platform-independent "tsWxGTUI" Toolkit (performs character-mode terminal input/output via an emulated subset of the high-level, pixel-mode "wxWidgets"/"wxPython" API)
- 4 **Low-Level Terminal Interface Library** - The system-specific "nCurses" Toolkit (performs character-mode terminal input/output via the low-level character-mode terminal-independent API)

System Administrators, Software Engineers and System Operators may use a broad range of platform hardware and software configurations. Candidate configurations including or equivalent to the following:

Operating System Software	Add-On Software	Central Processing Unit Hardware
Linux (CentOS, Fedora, Mandriva, Red Hat, Scientific, SuSE, Ubuntu etc.)	<ul style="list-style-type: none"> ▪ Python 2.6.x ▪ Python 2.7.x ▪ Python 3.0.x ▪ Python 3.1.x ▪ Python 3.2.x ▪ Python 3.3.x 	<ul style="list-style-type: none"> ▪ Intel & AMD / PC Compatible (32-bit & 64-bit) ▪ Freescale / IBM / Motorola PowerPC Compatible (32-bit & 64-bit) ▪ IBM Cell Broadband Engine (32-bit & 64-bit)
Mac OS X (10.4.x-10.9.x etc.)	<ul style="list-style-type: none"> ▪ Python 2.6.x ▪ Python 2.7.x ▪ Python 3.0.x ▪ Python 3.1.x ▪ Python 3.2.x ▪ Python 3.3.x ▪ VMware Fusion 3 / 4 / 5 (runs various Linux, Solaris, Unix and Windows Virtual Machines on Mac OS X using shared computer processor, memory, peripheral, and network resources) ▪ Parallels Desktop 5 / 6 / 7 / 8 / 9 for Mac (runs various Linux and Windows Virtual Machines on Mac OS X using shared computer processor, memory, peripheral, and network resources) 	<ul style="list-style-type: none"> ▪ Intel & AMD / PC Compatible (32-bit & 64-bit) ▪ Freescale / IBM / Motorola PowerPC Compatible (32-bit & 64-bit)
Microsoft Windows (8.1, 8, 7, Vista, XP etc.)	<ul style="list-style-type: none"> ▪ Cygwin 1.7.x/1.8.x, Free, Linux-/UNIX-style plug-in operating environment for MS Windows ▪ Python 2.6.x ▪ Python 2.7.x ▪ Python 3.0.x ▪ Python 3.1.x ▪ Python 3.2.x ▪ Python 3.3.x 	<ul style="list-style-type: none"> ▪ Intel & AMD / PC Compatible (32-bit & 64-bit)

UNIX (FreeBSD, HP-UX, IBM-AIX, Oracle/Sun-Solaris, Silicon Graphics Incorporated-IRIX etc.)	<ul style="list-style-type: none"> ▪ Python 2.6.x ▪ Python 2.7.x ▪ Python 3.0.x ▪ Python 3.1.x ▪ Python 3.2.x ▪ Python 3.3.x 	<ul style="list-style-type: none"> ▪ Intel & AMD / PC Compatible (32-bit & 64-bit) ▪ Freescale PowerPC Compatible (32-bit & 64-bit) ▪ Oracle/Sun SPARC or Compatible (32-bit & 64-bit) ▪ Silicon Graphics Incorporated MIPS or Compatible (32-bit & 64-bit)
---	--	---

2.3.4 Forms, Procedures, or Other Manual Operations

This paragraph shall identify the forms, procedures, or other manual operations that must be present.

- **Forms** (on page 43)
- **Procedures** (on page 44)
- **Other Manual Operations** (on page 44)

2.3.4.1 Forms

1 Host Operating System Documentation

- a) Cygwin (1.7, 1.8 etc.)
- b) Linux (Fedora 20, Mandriva, Red Hat, Scientific/Centos 6.5, SuSE, Ubuntu 12.04 etc.)
- c) Mac OS X (10.4.x-10.9.x) etc.
- d) Microsoft Windows (8.1, 8, 7, Vista, XP etc.)
- e) Unix (FreeBSD/PC-BSD 10, HP-UX, IBM-AIX, Oracle/Sun-Solaris 11, Silicon Graphics Incorporated-IRIX etc.)

2 Python Tutorial

- a) Dive into Python
- b) Python Style Guide
- c) Python Distutils

3 Python Language Reference Manual and Global Module Index

- a) Python 2.6
- b) Python 2.7
- c) Python 3.0
- d) Python 3.1
- e) Python 3.2
- f) Python 3.3
- g) Python 3.4

- 4 "wxWidgets" Documentation
 - a) Application Programming Interface
 - b) Tutorial
- 5 "wxPython" Documentation
 - a) Application Programming Interface
 - b) Demonstration
 - c) Tutorial
- 6 "nCurses" Documentation
 - a) Release Notes
 - b) Tutorial

2.3.4.2 Procedures

TBD - Under Construction

2.3.4.3 Other Manual Operations

TBD - Under Construction

2.3.5 Other Facilities, Equipment, or Resources

This paragraph shall identify the other facilities, equipment, or resources that must be present.

None

2.4 Software Organization and Overview of Operation

This paragraph shall provide a brief description of the organization and operation of the software from the user's point of view.

- *Logical Components of the Software* (on page 45)
- *Performance Characteristics* (on page 60)
- *Relationship to Interfacing Systems* (on page 61)
- *Supervisory Controls* (on page 77)

2.4.1 Logical Components of the Software

This paragraph shall provide a brief description of the organization of the software from the user's point of view.

- 1 **Block Diagram** (on page 45) - High level depiction of the organizational, functional and interface relationship between the "tsWxGTUI" Toolkit components and users (System Administrator, Software Engineer, System Operator and Field Service personnel):
 - a) A Command Line Interface ("tsToolkitCLI") - It provides the foundation for the Graphical User Interface ("tsToolkitGUI").
 - b) A Graphical-style User Interface ("tsToolkitGUI") - It uses the services of the "tsToolkitCLI" when appropriate.
 - c) How "tsToolkitCLI" and tsToolkitGUI" collaborate and interface with an operator (System Administrator, Software Engineer, System Operator or Field Service personnel).
- 2 **Stand Alone System Architecture** (on page 46) - High level depiction and description of the components of and relationship between components of an isolated system operating by itself.
- 3 **Stand Among System Architecture** (on page 50) - High level depiction and description of the components of and relationship between two or more networked systems operating in collaboration with each other.

2.4.1.1 Block Diagram

This Block Diagram depicts the organizational, functional and interface relationship between the TeamSTARS "tsWxGTUI_PyVx" Toolkit components and users (System Administrator, Software Engineer, System Operator and Field Service personnel):

- 1 the external System Operator interface to "tsToolkitCLI"
- 2 the internal "tsToolkitCLI" interface to "tsToolkitGUI"
- 3 the internal System Operator interfaces:
 - a) to "tsUtilities", "tsToolsCLI" and "tsTestsCLI" via "tsToolkitCLI"
 - b) to "tsToolsGUI" and "tsTestsGUI" via "tsToolkitCLI" and "tsToolkitGUI"

Graphical-style User Interface (tsToolkitGUI)	
<ul style="list-style-type: none"> The "tsToolsGUI" is a set of graphical-style user interface application programs for tracking software development metrics. 	<ul style="list-style-type: none"> The "tsTestsGUI" is a set of graphical-style user interface application programs for regression testing and tutorial demos.
<ul style="list-style-type: none"> The "tsLibGUI" is a library of graphical-style user interface building blocks that establishes the emulated run time environment for the high-level, pixel-mode, "wxPython" GUI Toolkit via the low-level, character-mode, "nCurses" Text User Interface Toolkit. 	

^ ^ |
 | | |
 | | v

Command Line Interface (tsToolkitCLI)	
<ul style="list-style-type: none"> The "tsToolsCLI" is a set of command line interface application programs and utility scripts for: checking source code syntax and style via "plint"; generating Unix-style "man page" documentation from source code comments via "pydoc"; and installing, modifying for publication, and tracking software development metrics. 	<ul style="list-style-type: none"> The "tsTestsCLI" is a set of command line interface application programs and scripts for regression testing and tutorial demos.
<ul style="list-style-type: none"> The "tsLibCLI" is a library of command line building blocks that establishes the POSIX-style, run time environment for pre-processing source files, launching application programs, handling events (registering events with date, time and event severity annotations) and configuring console terminal and file system input and output. 	
<ul style="list-style-type: none"> The "tsUtilities" is a library of computer system configuration, diagnostic, installation, maintenance and performance tool components for various host hardware and software platforms. 	

^ ^ |
 | | +- > Operator Display & Log Files
 | +----- Operator Keyboard
 +----- Operator Mouse

2.4.1.2 Stand Alone System Architecture

The *Team*STARS "tsWxGTUI_PyVx" Toolkit provides:

1. Python version-specific components for its Command Line Interface ("tsToolkitCLI")
2. Python version-specific components for its Graphical-style User Interface ("tsToolkitGUI").

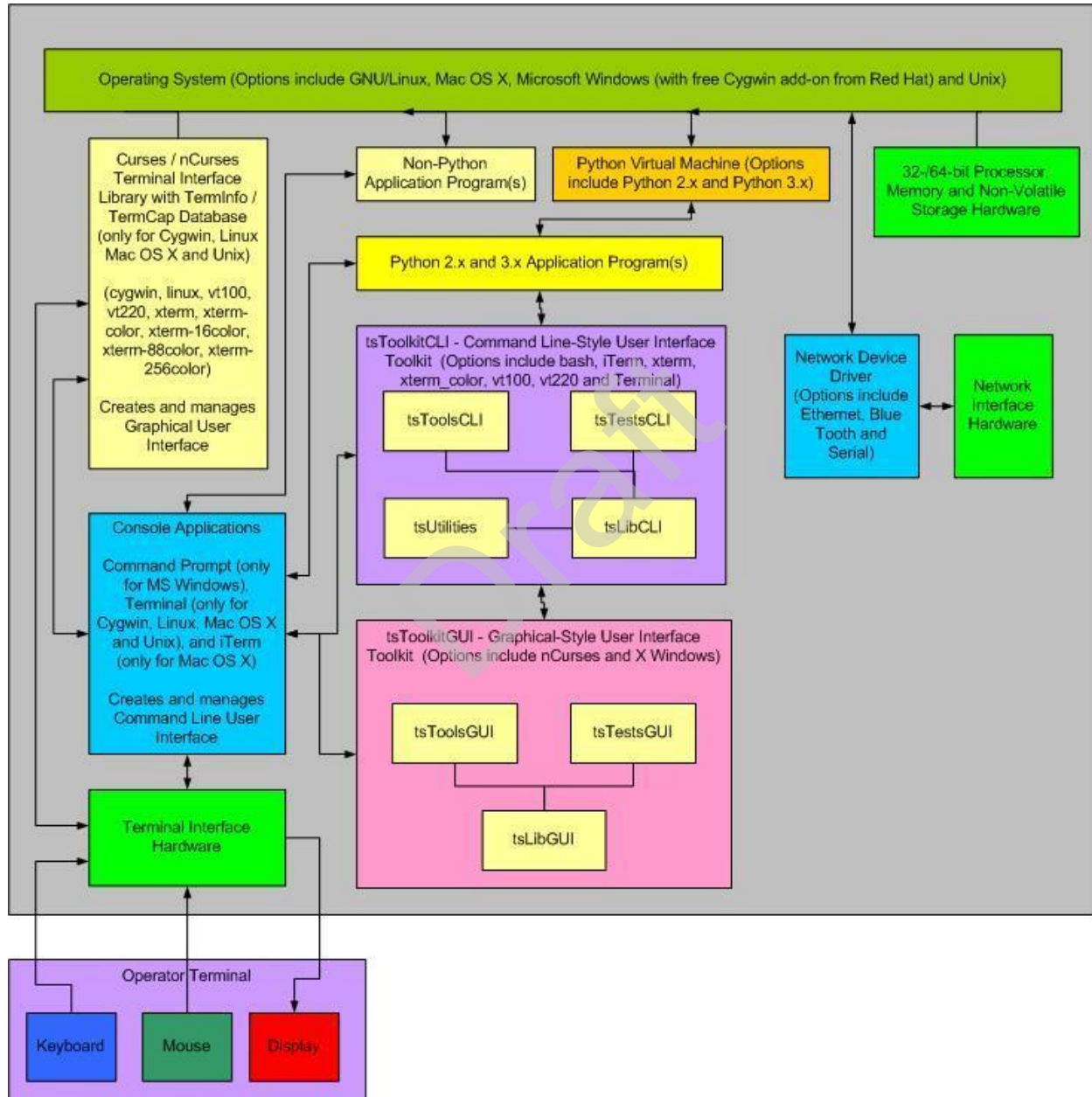
The following description uses the component names as depicted in the **Block Diagram** (on page 45)

This section depicts and describes the organization, function of and interface between various system hardware and software components and "tsWxGTUI_PyVx" Toolkit users (System Administrator, Software Engineer, System Operator and Field Service personnel):

- 1 the external System Operator interface to "tsToolkitCLI"
- 2 the internal "tsToolkitCLI" interface to "tsToolkitGUI"
- 3 the internal System Operator interfaces:

- a) to "tsLibCLI", "tsToolsCLI" . "tsTestsCLI" and "tsUtilities" via "tsToolkitCLI"
- b) to "tsLibGUI", "tsToolsGUI" and "tsTestsGUI" via "tsToolkitGUI" and "tsToolkitCLI"

This depiction represents a typical Stand Alone System configuration. In this configuration, the optional Network Hardware Interface and its associated Network Device Driver Interface should not be used, even if present, in order to avoid activities that adversely impact system performance.



- 1 **Operating System** - The platform specific software (such as Linux, MacOS X, Microsoft Windows and Unix) that coordinates and manages the time-shared use of a platform's processor, memory, storage and input/output hardware resources by multiple application programs and their associated users/operators.

2 Operator Terminal - A device for human interaction that includes:

- a) A Keyboard unit for text input
- b) A Mouse unit (mouse, trackball, trackpad or touchscreen with one or more physical or logical buttons) for selecting one of many displayed GUI objects to initiate an associated action.
- c) A Display unit (1-color "ON"/"OFF" or multi-color two-dimensional screen) for output of text and graphic-style, tiled and overlaid boxes.

3 Terminal Hardware Interface - The platform specific hardware with connections to the device units of the Operator Terminal.

- a) A PS/2 Port is a type of input port developed by IBM for connecting a mouse or keyboard to a Personal Computer. It supports a mini DIN plug containing just 6 pins.
- b) An RS-232C or RS-422 port for connecting a mouse for position and button-click input.
- c) A Universal Serial Bus (USB) port is an industry standard first developed in the mid-1990s that was designed to standardize the connection of computer peripherals (including keyboards, pointing devices, digital cameras, printers, portable media players, disk drives and network adapters) to personal computers, both to communicate and to supply electric power. It has effectively replaced a variety of earlier interfaces, such as serial and parallel ports, as well as separate power chargers for portable devices.

The USB 1.0 specification was introduced in January 1996. It defined data transfer rates of 1.5 Mbit/s "Low Speed" and 12 Mbit/s "Full Speed". The first widely used version of USB was 1.1 (now called "Full-Speed"), which was released in September 1998. Its 12 Mbit/s data rate was intended for higher-speed devices such as disk drives, and its lower 1.5 Mbit/s rate for low data rate devices such as joysticks.

The USB 2.0 (now called "Hi-Speed") specification was released in April 2000. It defined a higher data transfer rate, with the resulting specification achieving 480 Mbit/s, a 40-times increase over the original USB 1.1 specification.

The USB 3.0 specification (now called "SuperSpeed") was preleased in November 2008. It defined an even higher data transfer rate (up to 5 Gbit/s) and was backwards-compatible with USB 2.0. It added a new, higher speed bus called SuperSpeed in parallel with the USB 2.0 bus.

- d) A Video Adapter is a computer circuit card that provides digital-to-analog conversion, video RAM, and a video controller so that data can be sent to a computer's display. It typically adheres to the de facto standard, Video Graphics Array (VGA). VGA describes how data - essentially red, green, blue data streams - is passed between the computer and the display. It also describes the frame refresh rates in hertz. It also specifies the number and width of horizontal lines, which essentially amounts to specifying the resolution of the pixels that are created. VGA supports four different resolution settings and two related image refresh rates. The maximum VGA resolution setting produces a display that is 640 pixels wide by 480 pixels high. For a character font that is 8 pixels wide by 12 pixels high, the longest line of text will be 80 characters wide and there can be up to 40 lines of text displayed at any moment. Higher resolutions, such as SVGA, are supported by more advanced Video Adapters. The higher resolution settings typically require use of proportionally larger displays in order to maintain the size and legibility of the displayed text.

4 Terminal Device Driver - The platform specific software for transforming data (such as single button scan codes, multi-button flags and pointer position) to and from the platform independent formats (such as upper and lower case text, display screen column and row and displayed colors, fonts and special effects) used by the Command Line Interface and Graphical User Interface software.

- 5 **Command Line-Style Interface ("tsToolKitCLI")** - The platform specific keywords arguments, positional arguments and their associated values and syntax of text used to request services from the Operating System and various Application Programs.
 - a) **"tsLibCLI"** --- A library of command line building blocks that establishes the POSIX-/Unix-style, run time environment for pre-processing source files, launching application programs, handling events (registering events with date, time and event severity annotations) and configuring console terminal and file system input and output.
 - b) **"tsToolsCLI"** --- A set of command line interface application programs and utility scripts for: checking source code syntax and style via "plint"; generating Unix-style "man page" documentation from source code comments via "pydoc"; and installing, modifying for publication and tracking software development metrics.
 - c) **"tsTestsCLI"** --- A set of command line interface application programs and utility scripts for unit, integration and system level regression testing.
 - d) **"tsUtilities"** --- A library of computer system configuration, diagnostic, installation, maintenance and performance tool components for various host hardware and software platforms.
- 6 **Graphical-Style User Interface ("tsToolKitGUI")** - The platform specific tiled, overlaid and click-to-select Frames, Dialogs, Pull-down Menus, Buttons, CheckBoxes, Radio Buttons, Scrollbars and associated keywords, values and syntax of text used to request services from the Operating System and various Application Programs.
 - a) **"tsLibGUI"** --- A library of graphical-style user interface building blocks that establishes the emulated run time environment for the high-level, pixel-mode, "wxPython" GUI Toolkit via the low-level, character-mode, "nCurses" Text User Interface Toolkit.
 - b) **"tsToolsGUI"** --- A set of graphical-style user interface application programs for tracking software development metrics.
 - c) **"tsTestsGUI"** --- A set of graphical-style user interface application programs and command line interface utility scripts for unit, integration and system level regression testing.
- 7 **Python Application Program** - The application specific program that performs its service when executed by the Python Virtual Machine.
- 8 **Non-Python Application Program** - The application specific program that performs its service when its pre-compiled, platform specific machine code is executed. Typically, these services are used to analyze, edit, view, copy, move or delete those data and log files which are of interest or no longer needed.
- 9 **Python Virtual Machine** - The platform specific program that loads, interprets and executes the platform independent source code of a Python language application program.
- 10 **Processor, Memory, Storage and Communication Hardware** - Platform specific resources that are required by the Operating System and Application software.
- 11 **Network Hardware Interface** - The optional platform specific ethernet, blue-tooth and RS-232 serial port hardware for physical connections between the local system and one or more remote systems. It may also include such external hardware as gateways, routers, network bridges, switches, hubs, and repeaters. It may also include hybrid network devices such as multilayer switches, protocol converters, bridge routers, proxy servers, firewalls, network address translators, multiplexers, network interface controllers, wireless network interface controllers, modems, ISDN terminal adapters, line drivers, wireless access points, networking cables and other related hardware.

- a) An RJ-45 Ethernet port connecting to a local or wide area network. This port is capable of conducting simultaneous (full-duplex,) two-directional input and output at speeds over 100 gigabits per second. This port can also provide the interface to an optional printer shared with other network users.
- b) An RS-232C or RS-422 port for connecting a modem. Depending on the application, modems could be operated, at speeds over 56 kilobits per second, in either of two modes. In half duplex mode, each side alternated its sending and receiving roles. In full-duplex mode, each side could simultaneously send and receive.

12 Network Device Driver Interface - The optional platform specific software whose layered protocol suite (such as TCP/IP) enables the concurrent sharing of the physical connection between the local system and one or more remote systems.

NOTE: Concurrent local and remote login sessions are a convenience that must be used with caution. Time critical, resource intensive activities ought to be performed individually or sequentially (but NOT concurrently) on a Stand Alone System.

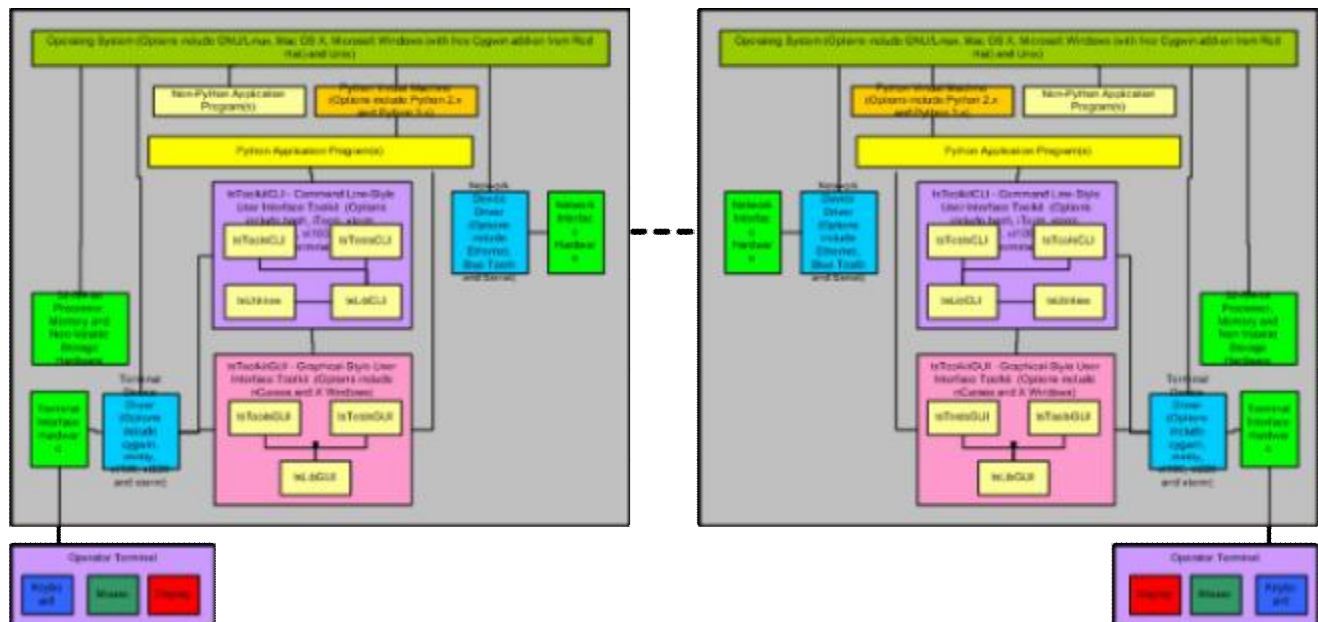
2.4.1.3 Stand Among System Architecture

The *Team*STARS "tsWxGTUI PyVx" Toolkit provides:

1. Python version-specific components for its Command Line Interface ("tsToolkitCLI")
2. Python version-specific components for its Graphical-style User Interface ("tsToolkitGUI").

The following description uses the component names as depicted in the **Block Diagram** (on page 45)

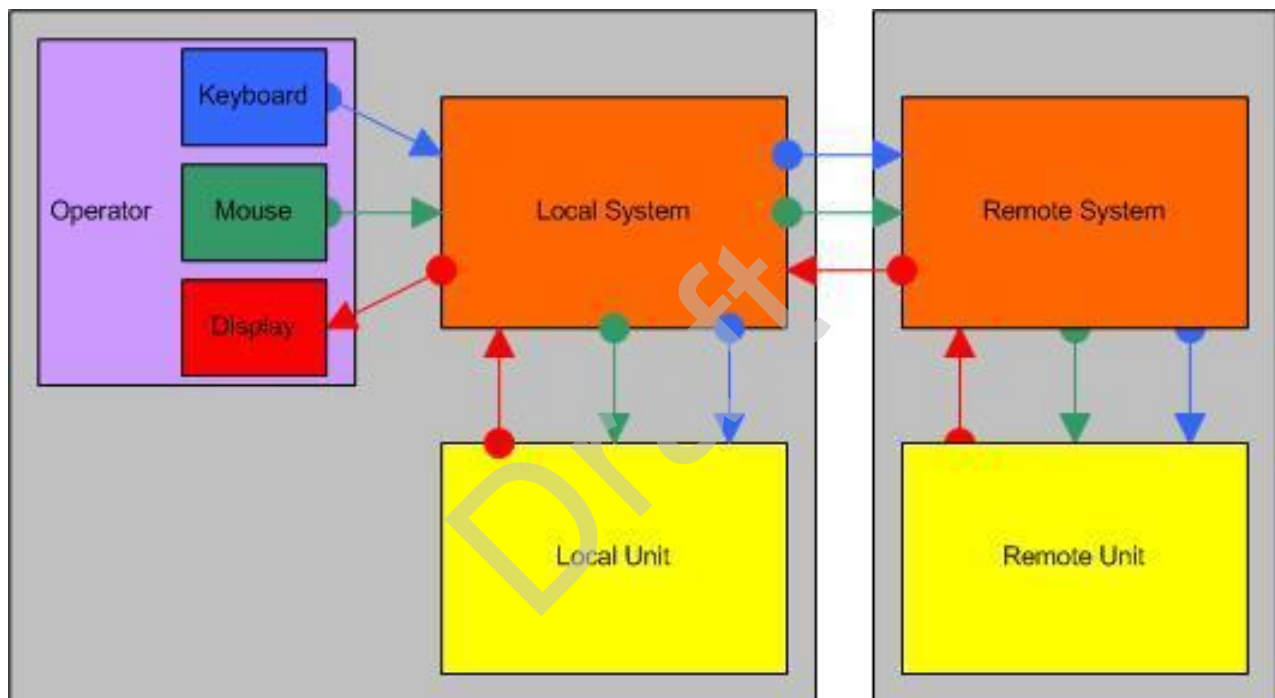
The ***Stand Alone System Architecture*** (on page 46), as previously described, may be extended to enable a single operator, working from a Local System, to interact with one or more Remote Systems.



In this configuration, the Local (Left) and Remote (Right) systems must first be networked via the available communication resources (Network Interface Hardware and Network Device Driver Interface).

Once networked, the local system operator must login to the Remote system via the "ssh user@Remote" command. The Local and Remote Terminal Device Interface then establishes a logical communication channel for exchanging keyboard, mouse and display information.

For each login Local and Remote session, the Operator may then select and run an Application Program. As depicted in the following figure. Application Programs run as Local Units on the system to which the Operator first logged in. Application Programs run as Remote Units on the systems to which the operator logged in via the "ssh user@Remote" command.



NOTE: Concurrent login sessions are a convenience that must be used with caution. Time critical, resource intensive activities ought to be performed individually or sequentially (but NOT concurrently) on a Stand Alone System. Only non-time critical, non-resource intensive activities may be performed concurrently on Stand Alone or Stand Among Systems.

- 1 **Local System (Left)** --- Operator opens one or more Command Line Interface Shells.
 - One or more newly opened shells may be used to run a Python or non-Python Application Program as its **Local Unit (Left.)**
 - One or more newly opened shells may be used to login to a Remote system (via the "ssh user@Remote" command). Once Remote login has been successful, the operator may run a Python or non-Python Application Program as a **Remote Unit (Right)**.
- 2 **Local Unit (Left)** --- A Python or non-Python Application Program that has been launched in a Local Command Line Interface Shell.
- 3 **Remote System (Right)** - Same Operator opens one or more Command Line Interface Shells.

- One or more newly opened shells may be used to run a Python or non-Python Application Program as its **Remote Unit (Right)**.

The **Multi-Session Desktop** (on page 52) figure depicts the desktop of a multi-user, multi-process, multi-threaded computer running the Professional Edition of Microsoft Windows 7. Among the background of desktop icons, there are two *TeamSTARS* "tsWxGTUI_PyVx" Toolkit sessions. The time each session displays synchronizes within its own one second refresh interval. The local session, on the left, is actively running Python 3.x on the Windows platform. The remote session, on the right, is actively running Python 2.x on the Mac OS X Yosemite platform which also serves as the Parallels 10 Hypervisor host for diverse Guest Operating Systems including:

Linux (CentOS7 64-bit, Fedora 21 64-bit, Scientific 6.5 32-bit and Ubuntu 12.04 32-bit)

Windows (98 SE 16-bit, XP 32-bit, 7 32-bit, 8 32-bit and 8.1 32-bit)

Unix (FreeBSD 9.2, PC-BSD 10, OpenIndiana 151a8 and OpenSolaris 11 32-bit)

- One or more newly opened shells may be used to login to a Remote system (via the "ssh user@Remote command). Once Remote login has been successful, the operator may run a Python or non-Python Application Program as a **Remote Unit (Right)**.

4 Remote Unit (Right) --- A Python or non-Python Application Program that has been launched in a Remote Command Line Interface Shell.

2.4.1.3.1 Multi-Session Desktop

From Wikipedia, the free encyclopedia

"In computer science, in particular networking, a session is a semi-permanent interactive information interchange, also known as a dialogue, a conversation or a meeting, between two or more communicating devices, or between a computer and user (see Login session). A session is set up or established at a certain point in time, and then torn down at some later point. An established communication session may involve more than one message in each direction. A session is typically, but not always, stateful, meaning that at least one of the communicating parts needs to save information about the session history in order to be able to communicate, as opposed to stateless communication, where the communication consists of independent requests with responses.

An established session is the basic requirement to perform a connection-oriented communication. A session also is the basic step to transmit in connectionless communication modes. However any unidirectional transmission does not define a session.[1]

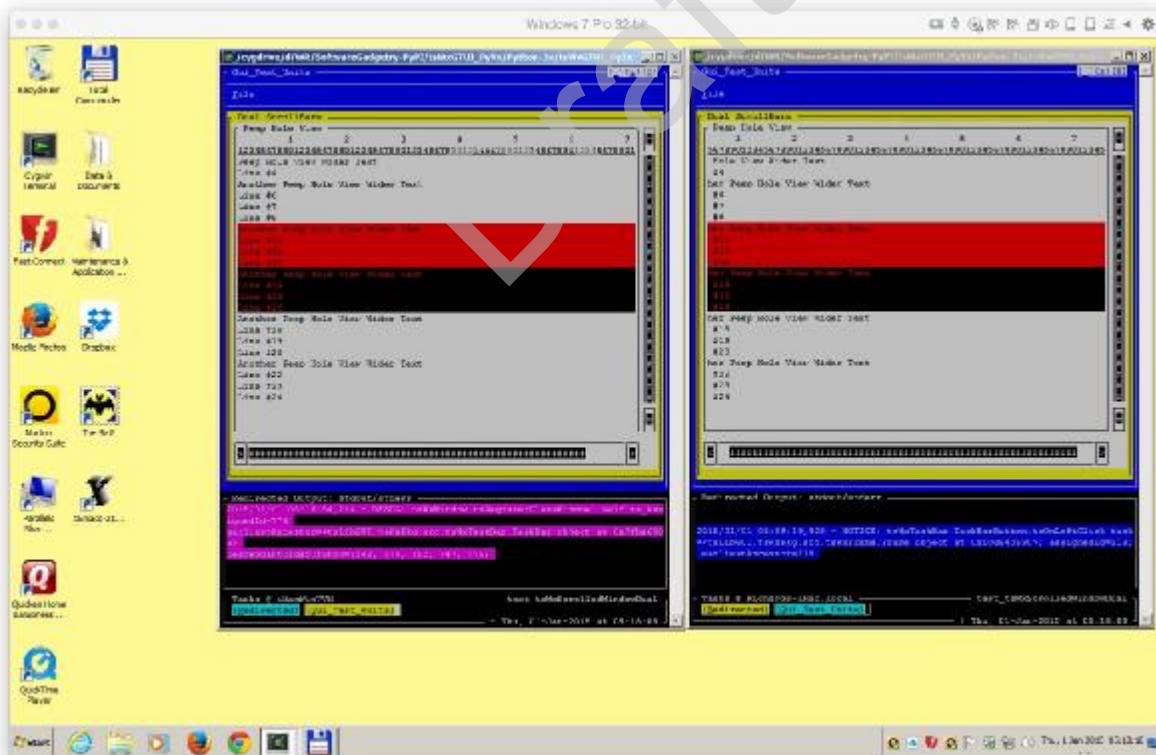
Communication sessions may be implemented as part of protocols and services at the application layer, at the session layer or at the transport layer in the OSI model.

- Application layer examples:
 - HTTP sessions, which allow associating information with individual visitors
 - A telnet remote login session
- Session layer example:
 - A Session Initiation Protocol (SIP) based Internet phone call
- Transport layer example:
 - A TCP session, which is synonymous to a TCP virtual circuit, a TCP connection, or an established TCP socket.

In the case of transport protocols that do not implement a formal session layer (e.g., UDP) or where sessions at the application layer are generally very short-lived (e.g., HTTP), sessions are maintained by a higher level program using a method defined in the data being exchanged. For example, an HTTP exchange between a browser and a remote host may include an HTTP cookie which identifies state, such as a unique session ID, information about the user's preferences or authorization level.

HTTP/1.0 was thought to only allow a single request and response during one Web/HTTP Session. However a workaround was created by David Hostettler Wain in 1996 such that it was possible to use session IDs to allow multiple phase Web Transaction Processing (TP) Systems (in ICL[disambiguation needed] nomenclature), with the first implementation being called Deity. Protocol version HTTP/1.1 further improved by completing the Common Gateway Interface (CGI) making it easier to maintain the Web Session and supporting HTTP cookies and file uploads.

Most client-server sessions are maintained by the transport layer - a single connection for a single session. However each transaction phase of a Web/HTTP session creates a separate connection. Maintaining session continuity between phases required a session ID. The session ID is embedded within the <A HREF> or <FORM> links of dynamic web pages so that it is passed back to the CGI. CGI then uses the session ID to ensure session continuity between transaction phases. One advantage of one connection-per-phase is that it works well over low bandwidth (modem) connections. Deity used a sessionID, screenID and actionID to simplify the design of multiple phase sessions."



The Sample Screenshot above shows a Windows 7 Desktop with:

- 1 A local Windows host with Python 3x session on left; and

2 A remote Mac OS X host with Python 2x session on right.

3 Each session consists of

- a) a wxPython-style Frame named "Dual ScrollBars" containing scrollable text with optional color markup.
- b) a wxPython-style Frame named "Redirected Output" containing date and time stamped event messages, with optional with color markup, that scroll up when new events are registered.
- c) a Host Desktop-style Frame named "Tasks @ Host Name" and Application Name with buttons to shift focus from background to foreground. There is also a spinner (to indicate the frequency or absence of idle time) and the current date and time (to indicate when the display was last updated).

2.4.1.4 tsDemoArchive

The "tsDemoArchive" directory contains an archival collection of known working application programs that demonstrate operational features of the "tsWxGTUI" Toolkit. The archival copies facilitate reverting experimental versions to the previously known working design.

The "tsWxGTUI" Toolkit is released with a copy of known working application programs installed in the top level directory ("tsWxGTUI").

The application programs in the "tsWxGTUI" directory may be modified in order to experiment with design and implementation alternatives.

2.4.1.5 tsDocumentation

The "tsDocumentation" directory contains the following subdirectories:

1 Development System Documents in PDF Format

- a) tsWxGTUI_Vol.__0_SDIST_Announcement
- b) tsWxGTUI_Vol.__1_SDIST_Brochure
- c) tsWxGTUI_Vol.__2_SDIST_Usage_Terms_&_Conditions
- d) tsWxGTUI_Vol.__3_SDIST_Introduction
- e) tsWxGTUI_Vol.__4_SDIST_Release_Notes
- f) tsWxGTUI_Vol.__5_SDIST_Software_Users_Manual

2 Embedded System Documents in ASCII Text Format

- a) Developer_Tutorial_Files
 - CLI_0_hello_world_print_statement.py
 - CLI_1_hello_world_print_function.py
 - CLI_2_hello_world_script_environment.py
 - CLI_3_hello_world_main_module_application.py
 - GUI_4_Curses_LowLevel_WidgetApi_application.py

GUI_5_SDIST_HighLevel_WidgetApi_application.py
GUI_6_SDIST_HighLevel_BoxSizerApi_application.py
ReadMe_Developer_Tutorial_Files.txt
test_tsCxBasics.py
test_tsWxBasics.py

b) Read_Me_Files

README.txt
README_1st.txt
README_1st-PyPI-Dev-tsWxGTUI.txt
README_1st-SoftwareGadgetry-Dev.txt
README-01-Title_Page.txt
README-02-Table_Of_Contents.txt
README-03-Purpose.txt
README-04-Goals.txt
README-05-Non-Goals.txt
README-06-Design_Strategy.txt
README-07-Design_Architecture.txt
README-08-Software_Configuration_Management.txt
README-09-tsWxGTUI_Directories.txt
README-10-tsToolkitCLI_Directories.txt
README-11-tsToolkitGUI_Directories.txt
README-12-Features.txt
README-13-Current_Capabilities.txt
README-14-Current_Limitations.txt
README-15-Reference_Documents.txt

c) Read_Me_Reference_Files

\$Table_Of_Contents.txt
AUTHORS.txt
BUGS.txt
CHANGE_LOG.txt
CONFIGURE.txt
COPYING.txt
CREDITS.txt
FAQ.txt

INSTALL.txt

MANIFEST.txt

NEWS.txt

OPERATE.txt

THANKS.txt

TO-DO.txt

TROUBLESHOOT.txt

d) Software_Project_Documents

README_1st-PyPI-Dev-tsWxGTUI.txt

README_1st-SoftwareGadgetry-Dev.txt

e) Usage_Terms_and_Conditions

COPY

RIGHT.txt

LICENSE.txt

NOTICES.txt

2.4.1.6 tsToolkitCLI

Command Line Interface ("tsToolkitCLI") - It provides the foundation for the Graphical User Interface ("tsToolkitGUI"). Its components include:

2.4.1.6.1 tsLibCLI

The library of building blocks is organized, by the functional scope of each component, into a collection of "packages". When appropriate, any package may import and use the services of any other tsLibCLI package.

Source code is contained in the associated ["src"] sub-directory. Depending on its complexity, the source code of a package may also be sub-divided and organized into a set of one or more source files.

Unit/Integration Test code is contained in the associated ["test"] sub-directory. Depending on its complexity, the test code of a package may also be sub-divided and organized into a set of one or more test files.

1 tsApplicationPkg - Base class to initialize and configure the application program launched by an operator. It enables an application launched via a Command Line Interface (CLI) to initialize, configure and use the same character-mode terminal with a Graphical-style User Interface (GUI).

Registers and validates instantiation settings input from applications via caller parameter list.

It also registers and validates operator application settings inputs from command line keyword-value pairs and positional arguments.

- 2 tsCommandLineEnvPkg** - Class to initialize and configure the application program launched by an operator. It delivers those keyword-value pair options and positional arguments specified by the application, in its invocation parameter list. It wraps the Command Line Interface application with exception handlers to control exit codes and messages that may be used to coordinate other application programs.

Establishes the command line environment for the application. Its services include platform configuration, initialization, input/output supervision and the application launcher.

- 3 tsCommandLineInterfacePkg** - Class establishes methods that prompt or re-prompt the operator for input, validate that the operator has supplied the expected number of inputs and that each is of the expected type.

- 4 tsCxGlobalsPkg** - Module to establish configuration constants and macro-type functions for the Command Line Interface mode of the "tsWxGTUI" Toolkit.

It provides a centralized mechanism for modifying/restoring those configuration constants that can be interogated at runtime by those software components having a "need-to-know". The intent being to avoid subsequent searches to locate and modify or restore a constant appropriate to the current configuration.

It also provides a theme-based mechanism for modifying/restoring those configuration constants as appropriate for various users and their activities.

- 5 tsDoubleLinkedListPkg** - Class to establish a representation of a linked list with forward and backward pointers.

- 6 tsExceptionPkg** - Class to define, categorize and handle error exceptions.

It wraps standard and application-specific error exception handlers to facilitate co-ordination of multiple application processes.

It maps run time exception types into 8-bit exit codes and prints associated diagnostic messages and traceback info.

- 7 tsLoggerPkg** - Class that emulates a subset of Python logging API.

It defines and handles prioritized, time and date stamped event message formatting and output to files and devices.

Files are organized in a date and time stamped directory named for the launched application.

Unix-type devices include syslog, stderr, stdout and stdscr (the ncurses display screen).

It also supports "wxPython"-style logging of assert and check case results.

- 8 tsOperatorSettingsParserPkg** - Class to parse the command line entered by the operator of an application program.

Parsing extracts the Keyword-Value pair Options and Positional Arguments that will configure and control the application during its execution.

Typical syntax:

```
python myApplication.py \
    [OPTION]... \
    [ARGUMENT]...
```

Typical examples:

```
./myApplication.py
python myApplication.py
python myApplication.py -h
python myApplication.py --help
python myFileCopier.py source.py target.py
```

The standard Python library module ("argparse", "optparse" or "getopt"), appropriate for the active Python version provides the basic parsing algorithm, unless the operator includes one of the following library module names ("optparse" or "getopt") as the last positional argument.

An application-specific configuration method provides the algorithm-specific definition of valid keyword-value pair options and/or positional arguments.

Upon input of an operator help request, the algorithm displays the expected GNU/POSIX command line syntax and usage help. Upon detecting an operator input syntax error, the algorithm displays an error message and usage help.

When used with Python 2.6-2.7 or Python 3.1-3.4, the "tsOperatorSettings.py" module (a "tsLibCLI" component of the "tsWXGTUI" Toolkit) supports the available Python version specific parser module(s) as an experimental and educational opportunity.

However, when one seeks to backport applications to Python 2.0-2.5 or Python 3.0, the "tsOperatorSettings.py" module ignores unsupported Python version specific parser modules in order to prevent a program trap which will block the application from running.

9 tsPlatformRunTimeEnvironmentPkg - Class to capture current hardware, software and network information about the run time environment for the user process.

This package tries to retrieve as much platform-identifying data as available. Its goal is to capture current hardware, software and network information about the run time environment for a user process. It makes this information available via a file (default is `"/PlatformRunTimeEnvironment.txt"`).

- a) Host processor hardware support includes various releases of Arm, x86, PowerPC, SPARC and others.
- b) Host operating system software support includes various releases of Cygwin, Linux ('debian', 'mandriva', 'redhat', 'SuSE', and 'Ubuntu'), Mac OS X, Unix ('FreeBSD', 'Solaris'), Windows ('8', '7', 'Vista', 'XP', '2000', '98') and others.
- c) Host virtual machine software support includes various releases of Java and Python.
- d) Network identification support includes host name, aliases and ip-address list.
- e) Environment Variable support includes user, session, shell, path and time zone.

10 tsReportUtilityPkg - Class defining methods used to format information: date and time (begin, end and elapsed), file size (with kilo-, mega-, giga-, tera-, peta-, exa-, zeta- and yotta-byte units) and nested Python dictionaries for display to an operator. It includes the following methods:

- a) `displayDictionary` - Recursive method to display nested entries in configuration dictionary.
- b) `getByteCountStrings` - Return a tuple with the text representations of a value.
- c) `getDateAndTimeString` - Construct timestamp string (in "Date at Time" format).

- d) `getDateTimeString` - Construct timestamp string suitable for use as label or alternatively for use as filename.
- e) `getDayHourMinuteSecondString` - Convert time from seconds to string format.
- f) `getDictionaryKeyTypeList` - Return a sorted list containing the key type in the specified dictionary.
- g) `getElapsedTimeString` - Construct elapsed time in days, hours, minutes, seconds between supplied startup and current time inputs in seconds since the UNIX epoch.
- h) `getHourMinuteSecondString` - Construct elapsed time in days, hours, minutes, seconds between supplied startup and current time inputs in seconds since the UNIX epoch.
- i) `getIndentString` - Construct a string of white space appropriate for indenting level.
- j) `getNextPathName` - Construct the path to the next log file.
- k) `getSecondsTimeFromHoursMinutesSecondsString` - Convert time from string to seconds format.
- l) `getSelectedDictionaryEntries` - Return a dictionary containing the specified type of entries from the specified dictionary.
- m) `getSeparatorString` - Construct a string of title and white space to separate one section of text from another.
- n) `getStatisticsList` - Create test summary after elapsed time and statistics details on the number of test runs, number of passing test runs, number of failing test runs, startup timestamp, shutdown timestamp and elapsed timestamp.
- o) `getTimeStatisticsList` - Generate Startup, Current (or Shutdown) and Elapsed Time strings.
- p) `getYearMonthDayString` - Construct date string (in "Year-Month-Day" format) from time in seconds since UNIX epoch.

11 `tsSysCommandsPkg` - Class definition and methods for issuing shell commands to and receiving responses from the host operating system.

It wraps and uses appropriate Python subprocess module methods.

2.4.1.6.2 `tsManPagesLibCLI`

POSIX-style User Manual Pages provide On-line Software Documentation, in text format, for `tsLibCLI` source files.

2.4.1.6.3 `tsManPagesToolsCLI`

POSIX-style User Manual Pages provide On-line Software Documentation, in text format, for `tsToolsCLI` source files.

2.4.1.6.4 `tsTestsCLI`

2.4.1.6.5 `tsToolsCLI`

2.4.1.6.6 `tsUtilities`

2.4.1.7 `tsToolkitGUI`

Graphical-style User Interface ("`tsToolkitGUI`") - It uses the services of the "`tsToolkitCLI`" when appropriate.

2.4.1.7.1 tsLibGUI

1 tsWxPkg - Collection of approximately 100 Classes that use the services of the Python Curses module to create a character-mode emulation of their pixel-mode "wxPython" Class counterparts. The collection includes widgets for frames, dialogs, panels, buttons, check boxes, radio boxes/buttons and scrollable text windows. It includes box and grid sizers. It also includes classes to emulate the host operating system theme-based color palette management, task bar, mouse click and window focus control services used/expected by "wxPython".

2.4.1.7.2 tsManPagesLibGUI

POSIX-style User Manual Pages provide On-line Software Documentation, in text format, for tsLibGUI source files.

2.4.1.7.3 tsManPagesToolsGUI

POSIX-style User Manual Pages provide On-line Software Documentation, in text format, for tsToolsCLI source files.

2.4.1.7.4 tsTestsGUI

2.4.1.7.5 tsToolsGUI

2.4.2 Performance Characteristics

Performance characteristics that can be expected by the user.

Item	Characteristic	Expected Performance
1	Types, volumes, rate of inputs accepted	
2	Types, volume, accuracy, rate of outputs that the software can produce	
3	Typical response time and factors that affect it	
4	Typical processing time and factors that affect it	
5	Limitations, such as number of events that can be tracked	
6	Error rate that can be expected	
7	Reliability that can be expected	

2.4.3 Relationship to Interfacing Systems

Relationship of the functions performed by the software with interfacing systems, organizations, or positions.

Item	Functions Performed	Interfacing Systems, Organizations or Positions
1	Python Module Import	<ul style="list-style-type: none"> Command Line Interface and Graphical-style User Interface building block component import from: Python (version-specific) Run Time Libraries on Local Host. Python (version-specific) Site Packages on Local Host. <p>This cross-platform interface will work without modification on all platforms.</p>
2	Operator Input	<p>Command Line Interface and Graphical-style User Interface input from:</p> <ul style="list-style-type: none"> Terminal/Console Keyboard, TrackBall, TrackPad and TouchScreen on Local and/or Remote Host (via appropriate operating system device drivers and terminal control libraries). Redirected input via stdinp. <p>This cross-platform interface will work without modification on all platforms.</p>
3	Operator Output	<ul style="list-style-type: none"> Command Line Interface and Graphical-style User Interface output to: Terminal/Console Display on Local and/or Remote Host (via appropriate operating system device drivers and terminal control libraries). Redirected output via stderr and stdout. <p>This cross-platform interface will work without modification on all platforms.</p>
4	File Input	<p>Command Line Interface and Graphical-style User Interface input from:</p> <ul style="list-style-type: none"> File System Object with Read Access on Local and/or Remote Host (via appropriate operating system device drivers). <p>This cross-platform interface will work without modification on all platforms.</p>
5	File Output	<p>Command Line Interface and Graphical-style User Interface output to:</p> <ul style="list-style-type: none"> File System Object with Append/Write Access on Local and/or Remote Host (via appropriate operating system device drivers). <p>This cross-platform interface will work without modification on all platforms.</p>
6	Application Launch	<p>Command Line Interface and Graphical-style User Interface application execution launch from:</p> <ul style="list-style-type: none"> File System Object with Execute/Read Access on Local and/or Remote Host (via appropriate operating system services). <p>This cross-platform interface will work without modification on all platforms.</p>
7	Event Logging	<p>Command Line Interface and Graphical-style User Interface output (annotated with date, time, severity level and source) to:</p> <ul style="list-style-type: none"> Terminal/Console Display on Local and/or Remote Host (via appropriate application-specific, Python and operating system services). File System Object with Append/Write Access on Local and/or Remote Host

		<p>(via appropriate application-specific, Python and operating system services).</p> <p>This cross-platform interface will work without modification on all platforms with the exception of the platform-specific services "syslog" (which is available only on Linux and Unix) and "eventlog" (which is available only on Microsoft Windows NT, 2000, XP and Vista).</p>
8	Exception Handling	<ul style="list-style-type: none"> Local and/or Remote Host (via appropriate application-specific, Python and operating system services). <p>This cross-platform interface will work without modification on all platforms.</p>
9	Command Line Interface (CLI)	<p>Command Line Interface has a Linux-style look and feel reminiscent of those user-friendly interfaces used in non-graphical, character-mode shells on Linux, Mac OS X, Microsoft Windows and Unix systems.</p> <p>Typically, the System Operator enters commands composed of alphabetic, numeric and punctuation characters from a keyboard. A command line parser module then extracts those Keyword-Value pair Options and Positional Arguments that will configure and control the application during its execution.</p> <p>Python version-specific parser modules available to the application include:</p> <ul style="list-style-type: none"> agparse (introduced with Python 2.7) optparse (introduced with Python 2.3) getopt (introduced with Python 1.6) <p>This cross-platform interface will work without modification on all platforms.</p>
10	Graphical-style User Interface (GUI)	<p>Graphical-style User Interface has a Linux-style look and feel reminiscent of those user-friendly interfaces used in graphical, pixel-mode shells on Linux, Mac OS X, Microsoft Windows and Unix systems.</p> <p>Typically, the System Operator enters command by clicking on GUI objects (buttons, checkboxes, radio boxes/buttons, scrollbars, sliders, task bar etc.). A window manager responds by appropriately modifying the display and updating it on an event-driven basis.</p> <p>The "tsWxGTUI" Toolkit window manager, acting on behalf of the application, uses Python's Curses module to create a character-mode emulation of the graphical, pixel-mode display of applications designed for the popular "wxWidgets" Toolkit and its "wxPython" binding for Python program.</p> <p>This cross-platform interface will work without modification on all platforms.</p>
11	Application Programming Interface (API)	<ul style="list-style-type: none"> "tsWxGTUI" Toolkit API "wxWidgets" API "wxPython" API "Python" API "Python Curses" API <p>This cross-platform interface will work without modification on all platforms.</p>

2.4.3.1 Design Strategy

The "tsWxGTUI" Toolkit's architecture and implementation reflects its previously described purpose, goals and non-goals. It incorporates the following popular, field-proven, free, open source technology which is comparable to their Commercial-Off-The-Shelf (COTS) product counterparts:

- 1 **"Python" Technology** (on page 63)
 - a) **Python Module Boiler Plate & Style** (on page 64)
 - b) **Python Module Import Technology** (on page 67)
 - c) **Python Module Launch Technology** (on page 69)
- 2 **"wxWidgets"/"wxPython" Technology** (on page 71)
- 3 **"nCurses" Technology** (on page 72)
- 4 **"tsWxGTUI" Toolkit Technology** (on page 72)
 - a) **Application Desktop Layout** (on page 75)

2.4.3.1.1 "Python" Technology

"Python" is a general-purpose, high-level object-oriented programming language whose design philosophy emphasizes code readability. It combines "remarkable power with very clear syntax", and its standard library is large and comprehensive. There is an extensive assortment of free, open source libraries from third-parties.

This technology supports legacy and present Python 2.x versions and present and future Python 3.x versions:

- 1 Enables the embedded system application programmer to design processor-independent applications that run, with little or no modification, on platforms with 32-bit and 64-bit processor architectures under "Linux" (Fedora, Mandriva, Red Hat, SuSE and Ubuntu), "Mac OS X" (10.5 - 10.9), "Microsoft Windows" (8.1, 8, 7, Vista and XP which all require "Cygwin", the free, Linux-like command line interface and GNU tools from Red Hat) and "Unix"-type operating systems (FreeBSD, HP-UX, IBM-AIX, SGI-IRIX and Sun/Oracle Solaris).
- 2 Enables the embedded system application programmer to design terminal-independent applications that run, with little or no modification, on platforms with the pixel-mode and character-mode terminals and terminal emulators available on the local and remote computer platforms.
- 3 Python's "Curses" module interfaces to the low-level, character-mode, "nCurses" GUI-style toolkit, the defacto standard for portable, platform-independant advanced terminal handling.
- 4 Converts positioning and sizing dimensions between the pixel-mode units used by "wxPython" applications and the character-mode units used by "Curses".
- 5 Converts "wxPython"-style color palettes and font attributes to their "Curses" equivalents.
- 6 Segregates Python-3.x from Python-2.x implementations because the 3.x branch has syntactic (try-except statement), symantic (handling of byte arrays) and library improvements that are not backward compatible.

NOTES:

1. Where practical, various Python-3.x features (print function) have been ported backwards to Python-2.6 and Python-2.7. The "tsWXGTUI" Toolkit takes advantage of this but it is still necessary to manually debug and fix the Python-3.x modules created by the Python "2to3" conversion utility.
 2. A few Python-3.x features have been made accessible to Python-2.5 via the "from __future__ import" mechanism. Alas, the "tsWXGTUI" Toolkit has not yet been designed to accomodate older Python versions.
-

2.4.3.1.1 Python Module Boiler Plate & Style

Boiler plate is an integral part of the "tsWxGTUI" Toolkit source code. It keeps developers and maintainers aware of design decisions, unintended consequences and previous corrective actions. External backup documentation, including check-in comments for source code version control systems, may become corrupt or totally lost.

The "tsWxGTUI" Toolkit provides its "tsStripComments.py" utility to remove boiler plate annotations inorder to limit memory usage in embedded systems and to conceal proprietary information from end-users and competitors.

The "tsWxGTUI" Toolkit uses the following boiler plate and style convention for its source code.

Boiler plate (text strings pre-formatted with embedded tab and new line characters), from the topmost module, is passed along for the following uses:

1 Intellectual Property Identification:

- a) `__title__` - Module name.
- b) `__version__` - Module **Software Versioning** (http://en.wikipedia.org/wiki/Software_versioning) in Major.Minor.BugFix (ex. 1.0.0, 1.0.70, 1.1.0 or Major.Minor.ReleaseType (ex. 1.0.a1, 1.1.b2, 1.2.rc3) notation.
- c) `__date__` - Module most recent update day/month/year (ex. 10/31/2014)
- d) `__authors__` - Module creator name(s) (ex. Frederick A. Kier and Richard S. Gordon)
- e) `__copyright__` - Module copyright notice
(ex. "Copyright (c) 2007-2013 ' + \n\n\t\tAll rights reserved.' % __authors__)
- f) `__license__` - Module terms and conditions notice.
(ex. 'GNU General Public License, ' + \n\n\t\tVersion 3, 29 June 2007')
- g) `__credits__` - Module third-party contributor name(s), copyright(s) and license(s)
(ex. '\n\n\t\tCredits: ' + \n\n\t\t\ttsLibCLI Import & Application Launch Features: ' + \

'\n\t Copyright (c) 2007-2009 Frederick A. Kier. ' + \

'\n\t\t\tAll rights reserved.')

h) __line1__ (a, b and c) - Module About & Launch header (title, version and date)

(ex. '%s, v%s (build %s)' % (__title__, __version__, __date__))

i) __line2__ (d) - Module About header (authors)

(ex. 'Author(s): %s' % __authors__)

j) __line3__ (e) - Module About header (copyright)

(ex. '%s' % __copyright__)

k) __line4__ (f, g) - Module About header (license and credits)

(ex. '%s%s' % (__license__, __credits__))

l) __header__ (h, i, j and k) - Module About header

(ex. '\n\n%s\n\n %s\n %s\n %s\n' % (__line1__,

__line2__,

__line3__,

__line4__))

m) mainTitleVersionDate (h) - Module Launch header

((ex. __line1__)

2 Source Code Documentation

a) Module Boiler Plate

Purpose - Statement of module's scope and objectives.

Capabilities - Statement of module's functional and interface contributions

Limitations - Statement of module's functional and interface short comings

Notes - Statement of module's design considerations

Usage (examples) - Statement of how library module is imported or how application module is invokled.

Classes - List and overview of available classes within module.

Methods - List and overview of available functions and methods within module.

Modifications - Chronological list and overview of module development and maintenance contributors and activities.

ToDo - Chronological list and overview of unresolved module functional and interface issues.

b) Class, Function and Method Comments - Docstring using PEP-257 Conventions

c) Algorithm Annotations - One or more # prefixed comment lines

d) Statement Annotations - One or more # prefixed comment lines

3 Naming Conventions

a) Graphical-style User Interface ("wxPython" GUI emulation)

"tsWx" prefix used for "wxPython" classes; tsWx class emulates wx class by importing all "tsWx" prefix classes and then aliasing back to "wxPython"-style "wx" prefix class names

"ts_" prefix used for "wxPython"-style "m_" prefix internal data

"ts" prefix used for non-"wxPython"-style functions and methods

b) Command Line Interface (CLI)

"ts" prefix used for classes, functions and methods

"ts_" prefix used for internal data

c) From the Python PEP 8 -- Style Guide for Python Code (<http://www.python.org/dev/peps/pep-0008/>):

"the following special forms using leading or trailing underscores are recognized (these can generally be combined with any case convention):

`_single_leading_underscore`: weak "internal use" indicator. E.g. "from M import *" does not import objects whose name starts with an underscore.

`single_trailing_underscore_`: used by convention to avoid conflicts with Python keyword, e.g.

`Tkinter.Toplevel(master, class_='ClassName')`

`__double_leading_underscore`: when naming a class attribute, invokes name mangling (inside class `FooBar`, `__boo` becomes `_FooBar__boo`; see below).

`__double_leading_and_trailing_underscore__`: "magic" objects or attributes that live in user-controlled namespaces. E.g. `__init__`, `__import__` or `__file__`. Never invent such names; only use them as documented.

Note that names with double leading and trailing underscores are essentially reserved for Python itself: "Never invent such names; only use them as documented".

4 Python Startup Module Settings:

a) Operator Settings.

Key-Word-Value Pair(s) and Positional Argument(s) extracted from the command line.

b) Application Launch Parameters.

Key-Word-Value Pair(s) and Positional Argument(s) extracted from the parameter list of the launched module.

5 System Operator Help:

a) About (-a, --about) - Displays information about the module and exits. Information includes module title, version, date, author(s), credit(s), copyright(s), license(s) and support contacts.

b) Help (-h, --help) - Displays module help and exits. Help includes command line syntax, usage and examples

c) Version(-v, --version) - Displays module version and exits.

2.4.3.1.1.2 Python Module Import Technology

Traditionally, a complex source code distribution organizes its packages within a common root directory which needs no `__init__.py` module. Each packages is then organized to contain any of its associated modules and an `__init__.py` module which may or not contain any logic.

By contrast, the "tsWxGTUI" Toolkit uses a non-traditional mechanism which facilitate the design, coding, debugging and non-duplicating import of building block library packages.

- 1 The "tsLibCLI" and "tsLibGUI" building block libraries each contain an `__init__.py` that identifies the associated packages in order of their dependancy.
 - a) Each package contains an `__init__.py` that is empty.
 - b) Each package must contain a "src" directory that contains two or more modules. The `__init__.py` identifies the package's ancestor (the library) and lists the package's modules in order of their dependancy.
 - c) Each package may contain a "test" directory that contains two or more modules. The `__init__.py` identifies the package's ancestor (the library) and lists the package's modules in order of their dependancy.
- 2 The "tsToolsCLI" and "tsToolsGUI" development tool libraries each contain an `__init__.py` that identifies the associated packages in order of their dependancy.
 - a) Each package contains an `__init__.py` that is empty.
 - b) Each package must contain a "src" directory that contains two or more modules. The `__init__.py` identifies the package's ancestor (the library) and lists the package's modules in order of their dependancy.
 - c) Each package may contain a "test" directory that contains two or more modules. The `__init__.py` identifies the package's ancestor (the library) and lists the package's modules in order of their dependancy.
- 3 Each application and "tsWxGTUI" Toolkit module only need to import those libraries it needs followed by those modules it needs. It need NOT specify such hierarchy details as `import tsLibCLI..tsLoggerPkg.tsLogger.src`. The collection of `__init__.py` modules collaborate in filling in the hierarchy details during the import process.
- 4 The following code fragment illustrates an explicit way the "tsWxGTUI" Toolkit handles the import dependency of the GUI features on the CLI ones:

```
__title__ = 'YourApplicationName'

GUImode = False
CLImode = not GUImode

# Python-based CLI Mode
try:
    import tsLibCLI
    import tsExceptions as tse
    import tsLogger as Logger
    import tsOperatorSettingsParser
    from tsDoubleLinkedList \
        import DoubleLinkedList

    if CLImode:

        from tsCommandLineEnv \
            import CommandLineEnv

except ImportError, importCode:

    fmt1 = '%s: ImportError ' + \
        '(tsLibCLI); ' % __title__
    fmt2 = 'importCode=%s' % str(importCode)
    msg = fmt1 + fmt2
    print(msg)

    raise tse.PROGRAM_EXCEPTION(
        tse.APPLICATION_TRAP, msg)

if GUImode:

    # wxPython-style, nCurses-based GUI Mode
    try:

        import tsLibGUI

        import tsWx as wx
        from tsWxMultiFrameEnv import MultiFrameEnv

    except ImportError, importCode:

        fmt1 = '%s: ImportError ' + \
            '(tsLibGUI); ' % __title__
        fmt2 = 'importCode=%s' % str(importCode)
        msg = fmt1 + fmt2

        print(msg)

        raise tse.UserInterfaceException(
            tse.CHARACTER_GRAPHICS_NOT_AVAILABLE,
            msg)

Representative "tsWxGTUI" Toolkit package __init__.py modules:
```

```

CLI mode:
    tsLibCLI/__init__.py
    tsLibCLI/tsLoggerPkg/__init__.py
    tsLibCLI/tsLoggerPkg/src/__init__.py

```

```

GUI mode:
    tsLibGUI/__init__.py
    tsLibGUI/tsWxPkg/__init__.py
    tsLibGUI/tsWxPkg/src/__init__.py

```

Representative "tsWxGTUI" Toolkit applications:

```

CLI mode: tsToolsCLI/tsPlatformQueryPkg/src/
          tsPlatformQuery.py

```

```

GUI mode: tsLibGUI/tsWxPkg/test/
          test_tsWxMultiFrameEnv.py

```

2.4.3.1.1.3 Python Module Launch Technology

1 **"tsApplication"** is a base class for launching the application specified by an operator. It initializes and configures the application using the following keyword value pairs and positional arguments:

- a) Input provided on the command line by an operator. The command line uses a Unix-style, free-format to promote future enhancement and on-going maintenance.
- b) Input provided in the parameter list of the application's invocation of the class instantiation. The parameter list uses a Python-style free-format to promote future enhancement and on-going maintenance.

2 **"tsCommandlineEnv"** is a convenience package wrapping terminal keyboard input, video display scrolled text output, "tsLogger" and "tsException" services. It is a base class for "tsWxMultiFrameEnv".

3 **"tsWxMultiFrameEnv"** is a convenience package wrapping terminal keyboard & mouse input, video display row and column addressable, field-editable output, "tsLogger" and "tsException" services.

4 All applications (with Command Line Interface or Graphical-style User Interface) begin with the appropriate one of the following Command Line Interface Launch configuration item list

a) **Graphical-style User Interface Wrapper Class**

```

from tsWxMultiFrameEnv import MultiFrameEnv
myApp = MultiFrameEnv(

```

b) **Command Line Interface Wrapper Class**

```

from tsCommandLineEnv import CommandLineEnv
myApp = CommandLineEnv(

```

c) **Command Line Interface Base Class** (for "tsWxGTUI" Toolkit developers only)

```

import tsApplication as tsAPP
myApp = tsAPP.TsApplication(

```

5 All applications (with Command Line Interface or Graphical-style User Interface) continue with the following Command Line Interface Launch configuration item list:

- a) buildTitle=__title__,

- b) buildVersion=__version__,
- c) buildDate=__date__,
- d) buildAuthors=__authors__,
- e) buildCopyright=__copyright__,
- f) buildLicense=__license__,
- g) buildCredits=__credits__,
- h) buildTitleVersionDate=mainTitleVersionDate,
- i) buildHeader=__header__,
- j) buildPurpose=__doc__,

The buildPurpose parameter presumes that the top-most module's purpose will be detailed in a Python docstring. Python docstrings appear as a string literal, (not an expression), as the first statement following the definition of functions, methods, modules, and classes. Docstrings can be accessed by the `__doc__` attribute on objects.

A Python docstring is defined by the single or multi-line text contained within three single (') or double (") quote characters as in the examples "the "wxPython" purpose" and ""the module's purpose"".

To prevent this docstring from being removed by the "tsStripComments" utility, it is suggested that this docstring be assigned explicitly to the variable `__doc__` (as in `__doc__ = ""the purpose""` rather than implicitly via `""the purpose""`).

However, as of 2014/02/27 there are no examples of this "tsStripComments" defensive strategy actually being applied by the "tsWxGTUI" Toolkit.

6 Python version appropriate Command Line Interface module(s) may obtain application-specific or non-Application-specific Keyword-Value pair Options and Positional Arguments and associated command line help:

- a) "argparse" module - introduced with Python 2.7.0
- b) "optparse" module - introduced with Python 2.3.0
- c) "getopt" module - introduced with Python 1.6.0
- d) enableDefaultCommandLineParser=False # Disable unless True

NOTES: The "tsWxGTUI" Toolkit includes two examples of this functionality:

- 1) The "tsToolsCLI" directory includes the "tsLinesOfCodeProjectMetrics" which provides an example of an application-specific parser in its "tsLOCPMOperatorSettingsParser.py" module.
 - 2) The "tsLibCLI" directory includes the "tsOperatorSettingsParserPkg" which provides an example of a non-application-specific parser sufficient enough to generate responses to only the following command line keywords: `-a / --about`, `-h / --help` and `-v / --version`.
 - 3) Each of the above examples includes Python version-specific methods that illustrate how to maintain your application's look and feel across host computer platforms with the latest version of Python or back-port it to an older version of Python.
-

- 7 When appropriate, some applications also use the following Graphical-style User Interface Launch configuration item list:
- a) `guiMessageFilename=None`,
 - b) `guiMessageRedirect=True`,
 - c) `guiRequired=True`,
 - d) `guiTopLevelObject=Sample_Instance`,
 - e) `guiTopLevelObjectId=wx.ID_ANY`,
 - f) `guiTopLevelObjectName='Sample_Name'`,
 - g) `guiTopLevelObjectParent=None`,
 - h) `guiTopLevelObjectPosition=wx.DefaultPosition`,
 - i) `guiTopLevelObjectSize=wx.DefaultSize`,
 - j) `guiTopLevelObjectStatusBar=None`,
 - k) `guiTopLevelObjectStyle=wx.DEFAULT_FRAME_STYLE`,
 - l) `guiTopLevelObjectTitle='Sample_Title'`,
- 8 When appropriate, some applications also use the following event logging configuration item list:
- a) When customized logging is appropriate, some applications use the following application-specific Launch configuration item:
`logs=['1st-Non-Default', ..., 'Nth-Non-Default']`,
 - b) When default logging is appropriate, some applications use the following non-application-specific Launch configuration item:
`logs=[]`,
- 9 All applications, with Command Line Interface or with both Command Line and Graphical-style User Interfaces, wrapup their Configuration item list as follows:
- a) `runTimeEntryPoint=Sample_Main`)

2.4.3.1.2 "wxWidgets"/"wxPython" Technology

"wxWidgets" (formerly "wxWindows") is a toolkit for creating pixel-mode Graphical User Interfaces (GUIs) for cross-platform applications on 32-bit and 64-bit architectures. It uses its own GUI API with the host operating system GUI services to achieve the native look and feel of the cross-platform GUI applications.

"wxWidgets" enables GUI application programs to compile and run on various computer platforms with minimal or no code changes. It covers systems such as Microsoft Windows, Mac OS X, iOS, Linux/Unix (X11, Motif, and GTK+), OpenVMS, OS/2 and AmigaOS.

"wxWidgets" is implemented in the C++ programming language. There are "wxWidgets" wrappers, such as "wxPython", available for use by programmers working in Python or other programming languages.

- 1 "wxWidgets" contributes the platform-independent C++ GUI Application Programming Interface (API) that establishes the architectural, functional, interface and exception handling requirements of candidate GUI API components.

- 2 "wxPython" contributes programmer reference documentation, tests and sample applications for the Python-specific candidate GUI API components.

2.4.3.1.3 "nCurses" Technology

"nCurses" (new curses) is a programming library that provides an API which allows the programmer to write text-based user interfaces in a terminal-independent manner.

- 1 Its low-level, character-mode, "nCurses" GUI-style Application Programming Interface (API) allows the "tsWxGTUI" Toolkit programmer to write text-based user interfaces in a terminal-independent manner that run with those terminals and terminal emulators available on the local and remote computer platforms.
- 2 It optimizes screen changes, in order to reduce the latency experienced when using remote shells.
- 3 It supports the following terminals and terminal emulators:
 - a) vt100/vt220 (1-color phosphor [green, orange or white] with 2-color pair black-on-color and color-on-black) without mouse input
 - b) xterm (supports 8-color black, blue, cyan, green, magenta, red, white and yellow / 64-color pair) with mouse input
 - c) xterm-color (deprecated early version of xterm)
 - d) xterm-16color (16-color black, blue, cyan, gray, green, lime green, magenta, maroon, navy, olive, purple, red, silver, teal, white and yellow / 256-color pair) with mouse input
 - e) xterm-88color (supports 88-color palette / 7744-color pair settings) with mouse input
NOTE: "nCurses" presets xterm-88color for xterm-16color compatibility.
 - f) xterm-256color (supports 181-color based on color-pair limit / 32767-color pair settings) with mouse input
NOTE: "nCurses" presets xterm-256color for xterm-16color compatibility.

2.4.3.1.4 "tsWxGTUI" Toolkit Technology

The "tsWxGTUI" Toolkit uses the Python Curses and "nCurses" technology to establish a character-mode Graphical-style User Interface that emulates the pixel-mode "wxWidgets" / "wxPython" Graphical User Interface. Extending the "wxPython" API to facilitate internal character-mode operations, provides an "wxPython"-style API which allows the programmer to write text-based user interfaces in a terminal-independent manner.

- 1 It supports the following terminals and terminal emulators:

NOTE: The "tsWxGraphicalTextUserInterface" Module of the "tsWxGTUI" Toolkit emulates "wxPython" and similarly guarantees support (via color matching or mapping to available "nCurses" non-color or color palette) for the following 68-color palette:

'aquamarine', 'black', 'blue', 'blue violet', 'brown', 'cadet blue', 'coral', 'cornflower blue', 'cyan', 'dark gray', 'dark green', 'dark olive green', 'dark orchid', 'dark slate blue', 'dark slate gray', 'dark turquoise', 'dim gray', 'firebrick', 'forest green', 'gold', 'goldenrod', 'gray', 'green', 'green yellow', 'indian red', 'khaki', 'light blue', 'light gray', 'light steel blue', 'lime green', 'magenta', 'maroon', 'medium aquamarine', 'medium blue', 'medium forest green', 'medium goldenrod', 'medium orchid', 'medium sea green', 'medium slate blue', 'medium spring green', 'medium turquoise', 'medium violet red', 'midnight blue', 'navy', 'orange', 'orange red', 'orchid', 'pale green', 'pink', 'plum', 'purple', 'red', 'salmon', 'sea green', 'sienna', 'sky blue', 'slate blue', 'spring green', 'steel blue', 'tan', 'thistle', 'turquoise', 'violet', 'violet red', 'wheat', 'white', 'yellow' and 'yellow green'.

- a) vt100/vt220 (1-color phosphor [green, orange or white] with 2-color pair black-on-color and color-on-black) without mouse input
- b) xterm (supports 68-color "wxPython" Palette with mapping to available 8-color black, blue, cyan, green, magenta, red, white and yellow / 64-color pair) with mouse input
- c) xterm-color (deprecated early version of xterm)
- d) xterm-16color (supports 68-color "wxPython" Palette with mapping to available 16-color black, blue, cyan, gray, green, lime green, magenta, maroon, navy, olive, purple, red, silver, teal, white and yellow / 256-color pair) with mouse input
- e) xterm-88color (supports 68-color "wxPython" Palette with color matching for 71-color / 5041-color pair settings) with mouse input

NOTE: "nCurses" presets xterm-88color for xterm-16color compatibility. The 68-color "wxPython" Palette cannot override the preset.

- f) xterm-256color (supports and extends 68-color "wxPython" Palette with color matching for 140-color / 19600-color pair settings) with mouse input

NOTE: "nCurses" presets xterm-256color for xterm-16color compatibility. The 68-color "wxPython" Palette cannot override the preset. The additional color settings demonstrate the capability for the application to introduce its own customization. However, "nCurses" can support only a 181-color Palette (based on its 32767 color-pair limit).

- 2** The "tsWxGTUI" Toolkit uses available Python module capabilities (and associated Python Curses and "nCurses" ones) to emulate those Host Operating System window manager capabilities typically used or provided by "wxWidgets" / "wxPython":

- a) Single Document Interface (SDI)

The System Operator may launch one or more Host Operating System Command Line Interface shells. Each shell supports a Command Line Interface or Graphical-style User Interface which is independant of all others. Each may be independantly configured (sized and positioned with application-specific terminal / terminal emulator features) and operated.

The System Operator may launch a single Python application program within each shell. The application program must terminate before the System Operator may launch it or a different application program.

- b) Multiple Document Interface (MDI)

The "tsWxGTUI" Toolkit provides this Graphical User Interface feature in which multiple windows reside under a single parent window (the associated System Operator designated Host Operating System Command Line Interface shell). Such Multiple Document Interfaces allow child windows to embed other windows inside them as well, creating complex nested hierarchies.

Additional details are provided in *Application Desktop Layout* (on page 75)

c) Redirected Output

The "tsWxGTUI" Toolkit provides this Graphical User Interface feature in which time critical information that must be called to the System Operator's attention is output to the bottom of a reserved area of the display. The information includes the date and time of the notification, its severity level and a brief descriptive message. Since the size of the reserved area is limited, previous information scrolls up and may disappear off the top of the area as new information becomes available.

The Command Line Interface provides a log file that captures a chronological history of past and current notifications.

Additional details are provided in *Application Desktop Layout* (on page 75)

d) Taskbar

The "tsWxGTUI" Toolkit provides this Graphical User Interface feature in which the System Operator can monitor the health of the system and control the visibility of information in the non-overlapped foreground and the partially or fully overlapped background. The taskbar is a reserved area of the display on the bottom edge of the GUI desktop. It is used to monitor and control running applications. It displays the application name, activity indicators (spinning baton, date and time) and buttons that shift focus from one Frame or Dialog to another, thereby moving partially concealed background objects to the fully visible foreground.

Additional details are provided in *Application Desktop Layout* (on page 75)

e) Triggering Event Identification

The "tsWxGTUI" Toolkit provides this Graphical User Interface feature in which keyboard and mouse click events are converted from their "Curses" into their equivalent "wxPython" identifiers.

Going well beyond the capabilities of Python Curses and "nCurses", the "tsWxGTUI" Toolkit scans the stack of "wxPython" GUI objects and identifies which one was visible enough to trigger the event. It dispatches the event to the appropriate "wxPython" event handler.

The "tsWxEventLoop" Module analyzes the screen coordinates of the mouse at the time of the click relative to the position, size and parent-child-focus relationship of each GUI object. By deduction, it identifies which topmost GUI object triggered the mouse click event.

The area in the top section of the display is reserved for application objects that may include one or more top-level windows (Frames or Dialogs) with one or more lower-level windows (titles, buttons, borders, text etc.) any of which may or may not be overlapped. The stacking order for overlapping the earliest ancestor of the descendant children of each Frame and Dialog determines which is the topmost GUI object.

The area in the mid section of the display is reserved for the "Redirected Output" objects that may include one or more lower-level windows (titles, buttons, borders, text etc.) any of which may or may not be overlapped.

The area in the bottom section of the display is reserved for the "Taskbar" objects that may include one or more lower-level windows (titles, buttons, borders, text etc.) any of which may or may not be overlapped.

f) Triggering Event Notification

The "tsWxGTUI" Toolkit provides this Graphical User Interface feature in which keyboard and mouse click event notifications are reported to the Toolkit or application-designated Event Handler.

The "tsWxEvtHandler" Module provides an extra boolean parameter ("useSystemEventTable") to determine if event notifications are reported to the Toolkit (via "SystemEventTable") or application (via "UserEventTable").

2.4.3.1.4.1 Application Desktop Layout

The "tsWxGTUI_PyVx" Toolkit's "tsLibGUI" library includes the "tsWxPkg" which includes the "tsWxDisplay" class. The class manages the layout of its assigned Command Line Interface shell screen (W-Columns x H-Rows) so as to provide:

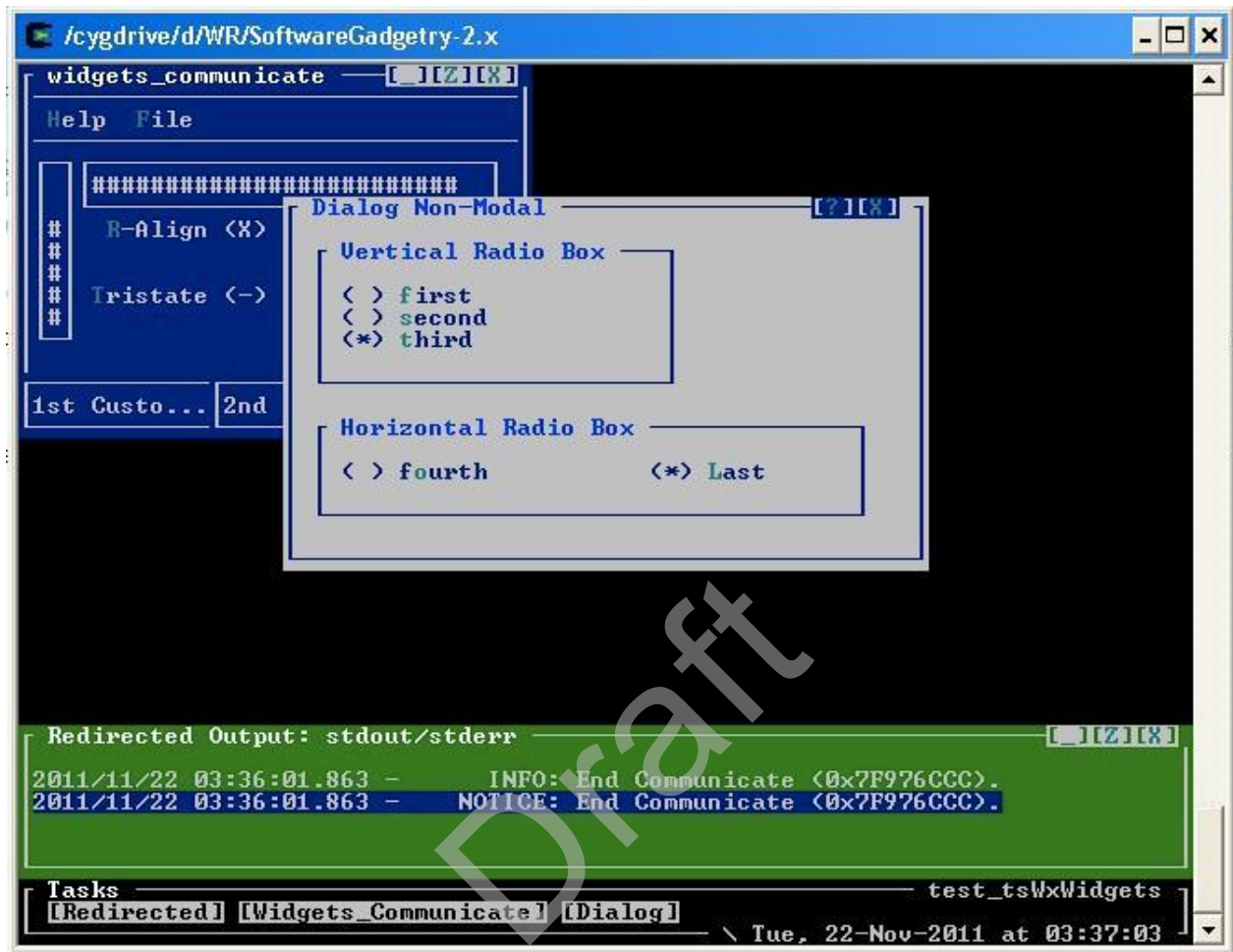
- 1 A client area for the application's Frame(s) and Dialog(s) windows.
- 2 An optional area for the application's auto-scrolling Redirected Output window.
- 3 An area for the application name, activity indicators (spinning baton, date and time) and buttons that shift focus from one Frame or Dialog to another, thereby moving partially concealed background objects to the fully visible foreground.

	X=0	1	2	3	X=W-1	
	0	1	2	3	4	
Y=0	+-----+ Frame & Dialog Border Top					
1	Client Area for an Application's					
2	Frame(s) & Dialog(s)					
3	(ex. 35 cols x 6 rows)					
4						
5	+-----+ Frame & Dialog Border Bottom					
H-9	+-----+ Stdio Frame Border Top					
H-8	Stdio Output (Optional)					
H-7	(ex. 35 cols x 5 rows)					
H-6						
H-5	+-----+ Stdio Frame Border Bottom					
H-4	+-----+ Task Frame Border Top					
H-3	Task Bar Output					
H-2	(ex. 35 cols x 4 rows)					
Y=H-1	+-----+ Task Frame Border Bottom					

The following screenshot illustrates the layout features:

- 1** The Command Line Interface was a "bash" shell created by "Cygwin" on a computer running Microsoft Windows XP.
- 2** The Graphical-style User Interface desktop was created by the "tsWxGTUI_PyVx" Toolkit.
- 3** The Application Program was named "test_tsWxWidgets" and was created for Python 2.7.
- 4** At the request of the Application Program, the "tsWxGTUI_PyVx" Toolkit (via its "tsWxFrame", "tsWxDialog" and "tsWxTxtCtrl" modules) created:
 - a) A client area frame named "widgets_communicate".
 - b) A client area dialog named "Dialog Non-Modal".
 - c) Logged a message with the INFO priority/severity level.
 - d) Logged a message with the NOTICE priority/severity level.
- 5** On behalf of the Application Program, the "tsWxGTUI_PyVx" Toolkit (via its "tsWxPyOnDemandOutputWindow" module) created:
 - a) A desktop frame named "Redirected Output: stdout/stderr".
 - b) A desktop frame named "Tasks" with:
 - Application name
 - Focus control buttons for two frames and one dialog
 - Periodically updated activity indicators

6 The screenshot was created on November 22, 2011 at 03:37:03 A.M.



2.4.4 Supervisory Controls

Supervisory controls that can be implemented (such as passwords) to manage the software.

Host computer systems with multi-user operating systems are equipped for System Administrators to establish login IDs and passwords on a per user basis.

Installed Python software and associated site-packages are typically shared by those users who successfully login.

There typically are no other Supervisor Controls.

2.5 Security and Privacy

This paragraph shall contain an overview of the security and privacy considerations associated with the software. A warning shall be included regarding making unauthorized copies of software or documents, if applicable.

- 1 The "tsWxGTUI" Toolkit, as authored and released by Richard S. Gordon a.k.a. Software Gadgetry, has NO involvement with the following security and privacy considerations:
 - a) System Administrators are responsible for establishing secure, personalized user accounts, passwords and permissions on those local and remote host computers to be used by the Software Engineer(s), System Operator(s) and Field Service Personnel.
 - b) Software Engineer(s), System Operator(s) and Field Service Personnel must first login to the local computer and then into any remote computer(s).
 - c) From Wikipedia, the free encyclopedia:

"A password is a word or string of characters used for user authentication to prove identity or access approval to gain access to a resource (example: an access code is a type of password), which should be kept secret from those not allowed access.

The use of passwords is known to be ancient. Sentries would challenge those wishing to enter an area or approaching it to supply a password or watchword, and would only allow a person or group to pass if they knew the password. In modern times, user names and passwords are commonly used by people during a log in process that controls access to protected computer operating systems, mobile phones, cable TV decoders, automated teller machines (ATMs), etc. A typical computer user has passwords for many purposes: logging into accounts, retrieving e-mail, accessing applications, databases, networks, web sites, and even reading the morning newspaper online.

A log in window for a website requesting a username and a password.

Despite the name, there is no need for passwords to be actual words; indeed passwords which are not actual words may be harder to guess, a desirable property. Some passwords are formed from multiple words and may more accurately be called a passphrase. The term passcode is sometimes used when the secret information is purely numeric, such as the personal identification number (PIN) commonly used for ATM access. Passwords are generally short enough to be easily memorized and typed.

Most organizations specify a password policy that sets requirements for the composition and usage of passwords, typically dictating minimum length, required categories (e.g. upper and lower case, numbers, and special characters), prohibited elements (e.g. own name, date of birth, address, telephone number). Some governments have national authentication frameworks[1] that define requirements for user authentication to government services, including requirements for passwords."

- 2 The "tsWxGTUI" Toolkit, as authored and released by Richard S. Gordon a.k.a. Software Gadgetry, has LIMITED involvement with the following security and privacy considerations:
 - a) The "tsWxGTUI" Toolkit can be installed for use by any previously established personalized user account(s).

- b) The "tsWxGTUI" Toolkit's "tsCxGlobals.py" and "tsWxGlobals.py" configuration files establish the input and output file locations on the designated local and remote host computer. Those file locations should only be changed by authorized System Administrator(s) and Software Engineer(s).
- c) The "tsWxGTUI" Toolkit only inputs information from the designated local and remote host computer keyboard, mouse and file systems.
- d) The "tsWxGTUI" Toolkit only outputs information to the designated local and remote host computer displays and files.

2.6 Assistance and Problem Reporting

This paragraph shall identify points of contact and procedures to be followed to obtain assistance and report problems encountered in using the software.

PROBLEM(s)	CONTACT	PROCEDURE(s)
Character-mode Cross-Platform "tsWxGTUI" CLI and GUI Toolkit Library, Tool and Utility Modules, Application Programming Interface and Documentation: <ul style="list-style-type: none"> ▪ Architecture ▪ Design ▪ Source Code ▪ QualificationTest ▪ Documentation 	Richard S. Gordon, a.k.a. Software Gadgetry, Principle Software Engineer, Author and Publisher <ul style="list-style-type: none"> ▪ WEB Site: www.SoftwareGadgetry.com ▪ E-Mail: Software Gadgetry@comcast.net 	<ul style="list-style-type: none"> ▪ Report your problems and findings to Software Gadgetry.

<p>Python Virtual Machine, Library Modules, Application Programming Interface and Documentation:</p> <ul style="list-style-type: none"> Python 2.x Python 3.x 	<p>Python Software Foundation</p> <ul style="list-style-type: none"> WEB Site: www.python.org 	<ul style="list-style-type: none"> Search the "Python" WEB Site for problem reports and new releases. Search your host computer operating system WEB Site for associated problem reports and new releases. Do NOT contact the Python Software Foundation. Report your problems and findings to Software Gadgetry.
<p>Pixel-mode Cross-Platform GUI Toolkit Library Modules, Application Programming Interface and Documentation:</p> <ul style="list-style-type: none"> "wxWidgets" "wxPython" 	<p>Julian Smart, Vadim Zeitlin, Stefan Csomor, Robert Roebling, and other members of the wxWidgets team.</p> <ul style="list-style-type: none"> WEB Site: www.wxwidgets.org 	<ul style="list-style-type: none"> Search the "wxWidgets" and "wxPython" WEB Sites for problem reports and new releases. Search your host computer operating system WEB Site for associated problem reports and new releases. Do NOT contact Julian Smart, Vadim Zeitlin, Stefan Csomor, Robert Roebling, and other members of the wxWidgets team. Report your problems and findings to Software Gadgetry.
<p>Character-mode, Terminal Control Library Modules, Application Programming Interface and Documentation:</p> <ul style="list-style-type: none"> "ncurses" 	<p>Thomas E. Dickey</p> <ul style="list-style-type: none"> WEB Site: http://invisible-island.net/ncurses/ 	<ul style="list-style-type: none"> Search the "ncurses" WEB Site for problem reports and new releases. Search your host computer operating system WEB Site for associated problem reports and new releases. Do NOT contact Thomas E. Dickey Report your problems and findings to Software Gadgetry.

3 ACCESS TO THE SOFTWARE

This section shall contain step-by-step procedures oriented to the first time/occasional user. Enough detail shall be presented so that the user can reliably access the software before learning the details of its functional capabilities. Safety precautions, marked by WARNING or CAUTION, shall be included where applicable.

- 1 **First-time User of the Software** (on page 82)
- 2 **Initiating a Session** (on page 92)
- 3 **Stopping and Suspending Work** (on page 92)

In the event you are accessing the software as part of a large organization which delegates responsibilities based on prior skills and current job description, the nature of the procedures have been categorized by the following functional roles:

- **System Administrator** (on page 109) - In this role, one or more individuals specify, plan, supervise and/or perform the installation, configuration, maintenance, repair and support of the computer hardware, operating system, application software and network to be used by Software Engineers and System Operators.
- **Software Engineer** (on page 110) - In this role, one or more individuals specify, plan, supervise and/or perform the design, development, coding, testing, debugging, integration, maintenance, release and support of application programs, with command line or graphical user interface, to be used by System Operators.
- **System Operator** (on page 111) - In this role, one or more individuals use various application programs, with associated command line or graphical user interfaces, to interactively supervise communication, control, data base, diagnostic, instrumentation or simulation activities.
- **Field Service Personnel** (on page 112) - In this role, one or more individuals install, repair and replace a manufacturer's hardware and software, at customer locations. They use diagnostic and maintenance programs, with associated command line or graphical user interfaces, to configure, test, tune and troubleshoot various hardware and software issues.

Otherwise, all of the previous functional roles will apply to you at one time or another. Its an opportunity to acquire new skills. Hopefully, the excerpted information from Wikipedia will suitably orient you.

3.1 First-time User of the Software

This paragraph shall be divided into the following subparagraphs.

- *Equipment Familiarization* (on page 82)
- *Access Control* (on page 91)
- *Installation and Setup* (on page 91)

3.1.1 Equipment Familiarization

This paragraph shall describe the following as appropriate:

- 1 *Computer System Startup* (on page 82)
- 2 *Computer System Terminal* (on page 83)
- 3 *Operator Terminal Usage* (on page 88)
- 4 *Computer System Shutdown* (on page 90)
 - *Procedures for turning on power and making adjustments* (see "*Computer System Startup*" on page 82) - Describes procedures for turning on power and making adjustments.
 - *Dimensions and capabilities of the visual display screen* (see "*Display*" on page 83) - Describes procedures for adjusting Dimensions and capabilities of the visual display screen.
 - *Appearance of the cursor, how to identify an active cursor if more than one cursor can appear, how to position a cursor, and how to use a cursor* (see "*Operator Terminal Usage*" on page 88) - Describes procedures for adjusting the appearance of the cursor, how to identify an active cursor if more than one cursor can appear, how to position a cursor, and how to use a cursor.
 - *Keyboard layout and role of different types of keys and pointing devices* (see "*Keyboard*" on page 85) - Describes Keyboard layout and role of different types of keys and pointing devices.
 - *Procedures for turning power off if special sequencing of operations is needed* (see "*Computer System Shutdown*" on page 90) - Describes procedures for turning power off if special sequencing of operations is needed.

3.1.1.1 Computer System Startup

This section describes procedures for turning on power and making adjustments.

Procedure to be followed by the System Administrator, Software Engineer, System Operator or Field Service Personnel.

- 1 Apply or restore electric power to the computer system and its associated independently powered peripherals.

See **Booting** (on page 100) for an overview of the process that automatically occurs within a host computer upon the application/restoration of electric power.

- 2 Adjust features (sizing, position, scrolling, font, colors and terminal emulation) of the visual display screen.

See *Dimensions and capabilities of the visual display screen* (see "*Display*" on page 83) for the applicable procedure.

3.1.1.2 Computer System Terminal

3.1.1.2.1 Display

This section describes dimensions, capabilities and adjustment of the visual display screen.

Host computer systems may be used in:

- 1 Embedded systems for monitoring and controlling equipment and application.

Such systems may have a small, low resolution terminal display screen, minimal resources (processing horse power, computer memory, communication and input/output capabilities).

Such systems may have an operating system that concurrently supports a limited number of local and remote users, multiple processes and multiple tasks (threads of execution), multiple applications and multiple services.

The terminal display screen may be configured and updated by the application.

- 2 General purpose systems used for software development.

Such systems may have large, high resolution terminal display screen, substantial resources (processing horse power, computer memory, communication and input/output capabilities).

Such systems may have an operating system that concurrently support numerous local and remote users, multiple processes and multiple tasks (threads of execution), multiple concurrent terminal emulator instances, multiple applications and multiple services.

Each terminal emulator screen area may be independently configured.

Procedure to be followed by the System Administrator, Software Engineer, System Operator or Field Service Personnel to adjust the visual display screen (size, position, scrolling, font, color palette, and terminal emulation), after Booting has been completed.

- 1 Linux

- a) Fedora Linux Terminal
- b) Scientific (CentOS) Linux Terminal
- c) Ubuntu Linux Terminal

- 2 Mac OS X

- a) Terminal
- b) iTerm2

- 3 Microsoft Windows

- a) Command Prompt

<http://www.microsoft.com/resources/documentation/windows/xp/all/proddocs/en-us/commandpromptoptions.mspix?mfr=true>

- b) Cygwin Mintty

<http://www.howtogeek.com/68511/how-to-improve-your-cygwin-experience-with-mintty/>

4 Unix

- a) OpenIndiana Terminal
- b) PC-BSD Terminal

3.1.1.2.1.1 Linux Display

3.1.1.2.1.2 Mac OS X Display

3.1.1.2.1.3 Microsoft Windows Display

From <http://www.microsoft.com/resources/documentation/windows/xp/all/proddocs/en-us/commandpromptoptions.mspx?mfr=true>

To Configure the Command Prompt

- 1** Open Command Prompt
- 2** Click the upper-left corner of the Command Prompt window, and then click Properties.
- 3** Click the Options tab.
- 4** In Command History, type or select 999 in Buffer Size, and then type or select 5 in Number of Buffers.
- 5** In Edit Options, select the Quick Edit Mode and Insert Mode check boxes.
- 6** Click the Layout tab.
- 7** In Screen Buffer Size, type or select 2500 in Height.
- 8** Do any of the following optional tasks:
 - In Screen Buffer Size, increase Width.
 - In Window Size, increase Height.
 - In Window Size, increase Width.
 - Clear the Let system position window check box, and then, in Window Position, change the values in Left and Top.
- 9** In the Apply Properties dialog box, click Save properties for future windows with same title.

3.1.1.2.1.4 Unix Display

3.1.1.2.2 Keyboard

This section describes keyboard layout and role of different types of keys and pointing devices.

From Wikipedia, the free encyclopedia:

"In computing, a keyboard is a typewriter-style device, which uses an arrangement of buttons or keys, to act as mechanical levers or electronic switches. Following the decline of punch cards and paper tape, interaction via teleprinter-style keyboards became the main input device for computers.

A keyboard typically has characters engraved or printed on the keys and each press of a key typically corresponds to a single written symbol. However, to produce some symbols requires pressing and holding several keys simultaneously or in sequence. While most keyboard keys produce letters, numbers or signs (characters), other keys or simultaneous key presses can produce actions or execute computer commands.

Despite the development of alternative input devices, such as the mouse, touchscreen, pen devices, character recognition and voice recognition, the keyboard remains the most commonly used and most versatile device used for direct (human) input into computers.[citation needed]

In normal usage, the keyboard is used to type text and numbers into a word processor, text editor or other programs. In a modern computer, the interpretation of key presses is generally left to the software. A computer keyboard distinguishes each physical key from every other and reports all key presses to the controlling software. Keyboards are also used for computer gaming, either with regular keyboards or by using keyboards with special gaming features, which can expedite frequently used keystroke combinations. A keyboard is also used to give commands to the operating system of a computer, such as Windows' Control-Alt-Delete combination, which brings up a task window or shuts down the machine. A command-line interface is a type of user interface operated entirely through a keyboard, or another device performing the function of one."

3.1.1.2.3 Mouse

This section describes the role of different types of pointing devices.

From Wikipedia, the free encyclopedia:

"Many mice have two buttons and a scroll wheel.

A computer mouse is an input device that is most often used with a personal computer. Moving a mouse along a flat surface can move the on-screen cursor to different items on the screen. Items can be moved or selected by pressing the mouse buttons (called clicking).[1]

It is called a computer mouse because of the wire that connects the mouse to the computer. The people who designed the first computer mice thought that it looked like the tail on a mouse. Today, many computer mice use wireless technology and have no wire.

History

When Silicon Valley was being reclaimed in California, Douglas Engelbart (1925-2013), a researcher of Stanford Research Institute, wanted to find a way to make using computers easier. In those days, computers were large and expensive. Using them was very hard because everything had to be typed in by hand, and there was no way to alter things if you made a mistake.

After studying and designing for a long time, Engelbart succeeded in inventing an input device which he named 'XY index'.

At first, it needed two hands to use, but it was changed so that only one hand was needed to use it. This model was more like the mouse that we use today. Xerox Palo Alto Research introduced a GUI in 1981, using a mouse.

The mouse was used with Macintosh[2] of Apple Inc. when it came out in 1984. Microsoft Windows also used the mouse when it came out, so over time computer mice became used with many computers. Modern mice have three buttons: left button, right button, scroll button.

Uses

On most computers, the user can move the mouse to move the cursor in the same direction.[3] To choose something that is on the screen, the user can move the cursor to it and "click" the left mouse button. The right mouse button is used to open menus that are different depending on where the cursor is. The other mouse buttons can do different things, depending on the software. Most mice have two or three buttons to click. Most mice also have a "scroll wheel"—a small wheel found between the two main mouse buttons. The user can move the wheel up or down to "scroll" through things like a website or folder, which means to move it up or down on the screen, or can click the wheel down like another button."

3.1.1.2.4 Trackball

From Wikipedia, the free encyclopedia:

"A trackball is a pointing device consisting of a ball held by a socket containing sensors to detect a rotation of the ball about two axes—like an upside-down mouse with an exposed protruding ball. The user rolls the ball with the thumb, fingers, or the palm of the hand to move a pointer.

Compared with a mouse, a trackball has no limits on effective travel; at times, a mouse can reach an edge of its working area while the operator still wishes to move the screen pointer farther. With a trackball, the operator just continues rolling, whereas a mouse would have to be lifted and re-positioned. Some trackballs, such as Logitech's optical-pickoff types, have notably low friction, as well as being dense (glass), so they can be spun to make them coast. The trackball's buttons may be situated to that of a mouse or to a unique style that suits the user.

Large trackballs are common on CAD workstations for easy precision. Before the advent of the touchpad, small trackballs were common on portable computers, where there may be no desk space on which to run a mouse. Some small thumbballs clip onto the side of the keyboard and have integral buttons with the same function as mouse buttons."

3.1.1.2.5 Touchscreen

From Wikipedia, the free encyclopedia:

"A touchscreen is an electronic visual display that the user can control through simple or multi-touch gestures by touching the screen with a special stylus/pen and-or one or more fingers. Some touchscreens use an ordinary or specially coated gloves to work while others use a special stylus/pen only. The user can use the touchscreen to react to what is displayed and to control how it is displayed (for example by zooming the text size).

The touchscreen enables the user to interact directly with what is displayed, rather than using a mouse, touchpad, or any other intermediate device (other than a stylus, which is optional for most modern touchscreens).

Touchscreens are common in devices such as game consoles, all-in-one computers, tablet computers, and smartphones. They can also be attached to computers or, as terminals, to networks. They also play a prominent role in the design of digital appliances such as personal digital assistants (PDAs), satellite navigation devices, mobile phones, and video games and some books (Electronic books).

The popularity of smartphones, tablets, and many types of information appliances is driving the demand and acceptance of common touchscreens for portable and functional electronics. Touchscreens are found in the medical field and in heavy industry, as well as for automated teller machines (ATMs), and kiosks such as museum displays or room automation, where keyboard and mouse systems do not allow a suitably intuitive, rapid, or accurate interaction by the user with the display's content.

Historically, the touchscreen sensor and its accompanying controller-based firmware have been made available by a wide array of after-market system integrators, and not by display, chip, or motherboard manufacturers. Display manufacturers and chip manufacturers worldwide have acknowledged the trend toward acceptance of touchscreens as a highly desirable user interface component and have begun to integrate touchscreens into the fundamental design of their products."

3.1.1.2.6 Touchpad / Trackpad

From Wikipedia, the free encyclopedia:

"A touchpad /ˈtʌtʃpæd/ or trackpad /ˈtrækpæd/ is a pointing device featuring a tactile sensor, a specialized surface that can translate the motion and position of a user's fingers to a relative position on the operating system that is outputted to the screen. Touchpads are a common feature of laptop computers, and are also used as a substitute for a mouse where desk space is scarce. Because they vary in size, they can also be found on personal digital assistants (PDAs) and some portable media players. Wireless touchpads are also available as detached accessories. In recent years the term "touchpad" is falling out of favor in exchange for "trackpad," as the term "touchpad" is often confused with tablet PCs or "convertible" touchscreen laptop computers.

Operation and function

Touchpads operate in one of several ways, including capacitive sensing and conductance sensing. The most common technology used as of 2010 entails sensing the capacitive virtual ground effect of a finger, or the capacitance between sensors. Capacitance-based touchpads will not sense the tip of a pencil or other similar implement. Gloved fingers may also be problematic.

While touchpads, like touchscreens, are able to sense absolute position, resolution is limited by their size. For common use as a pointer device, the dragging motion of a finger is translated into a finer, relative motion of the cursor on the output to the display on the operating system, analogous to the handling of a mouse that is lifted and put back on a surface. Hardware buttons equivalent to a standard mouse's left and right buttons are positioned below, above, or beside the touchpad.

Some touchpads and associated device driver software may interpret tapping the pad as a click, and a tap followed by a continuous pointing motion (a "click-and-a-half") can indicate dragging.[1] Tactile touchpads allow for clicking and dragging by incorporating button functionality into the surface of the touchpad itself.[2][3] To select, one presses down on the touchpad instead of a physical button. To drag, instead performing the "click-and-a-half" technique, one presses down while on the object, drags without releasing pressure and lets go when done. Touchpad drivers can also allow the use of multiple fingers to facilitate the other mouse buttons (commonly two-finger tapping for the center button).

Some touchpads have "hotspots", locations on the touchpad used for functionality beyond a mouse. For example, on certain touchpads, moving the finger along an edge of the touch pad will act as a scroll wheel, controlling the scrollbar and scrolling the window that has the focus vertically or horizontally. Many touchpads use two-finger dragging for scrolling. Also, some touchpad drivers support tap zones, regions where a tap will execute a function, for example, pausing a media player or launching an application. All of these functions are implemented in the touchpad device driver software, and can be disabled."

3.1.1.3 Operator Terminal Usage

This section describes the appearance of the cursor, how to identify an active cursor if more than one cursor can appear, how to position a cursor, and how to use a cursor.

1 Appearance

From Wikipedia, the free encyclopedia:

"In computing, a cursor is an indicator used to show the position on a computer monitor or other display device that will respond to input from a text input or pointing device. The flashing text cursor may be called a caret, as in caret browsing.[1] The mouse cursor is also called a pointer,[2] owing to its arrow shape on some systems."

2 How to Identify an Active Cursor

From Wikipedia, the free encyclopedia:

"In most command-line interfaces or text editors, the text cursor or caret navigation, is an underscore, a solid rectangle, or a vertical line, which may be flashing or steady, indicating where text will be placed when entered (the insertion point). In text mode displays, it was not possible to show a vertical bar between characters to show where the new text would be inserted, so an underscore or block cursor was used instead. In situations where a block was used, the block was usually created by inverting the pixels of the character using the boolean math exclusive or function.[3] On text editors and word processors of modern design on bitmapped displays, the vertical bar is typically used instead.

The blinking of the text cursor is usually temporarily suspended when it is being moved; otherwise, the cursor may change position when it is not visible, making its location difficult to follow.

Some interfaces use an underscore or thin vertical bar to indicate that the user is in insert mode, a mode where text will be inserted in the middle of the existing text, and a larger block to indicate that the user is in overtype mode, where inserted text will overwrite existing text. In this way, a block cursor may be seen as a piece of selected text one character wide, since typing will replace the text "in" the cursor with the new text."

3 How to Position a Cursor

a) Graphical User Interface (GUI)

Momentarily press the left (or only) mouse button to select the active window.

Move the mouse until the cursor is positioned at the desired input location.

Momentarily press the left (or only) mouse button to select the input location.

b) Command Line Interface (CLI)

Momentarily press the appropriate cursor positioning button on the keyboard:

"Home" button (or emacs-style CTRL-a) selects the left-most character on the current line.

"End" button (or emacs-style CTRL-e) selects the right-most character on the current line.

"Left" arrow button (or emacs-style CTRL-b) selects movement to the next character to left.

"Right" arrow button (or emacs-style CTRL-f) selects movement to the next character to right.

"Up" arrow button (or emacs-style CTRL-p) selects movement to up the previous line.

"Down" arrow button (or emacs-style CTRL-n) selects movement down to the next line.

4 How to Use a Cursor

Operator input may be created or modified after the cursor has been moved to an appropriate position.

Input may be inserted to the right of the cursor simply by pressing buttons (alphabetic, numeric, blank and punctuation) on the keyboard.

Input may be deleted to the right of the cursor simply by pressing the forward delete button (DELETE or emacs-style CTRL-d).

Input may be deleted to the left of the cursor simply by pressing the backward delete button (DELETE-BACKSPACE).

3.1.1.4 Computer System Shutdown

This section describes procedures for turning power off if special sequencing of operations is needed.

From Wikipedia, the free encyclopedia

"To shut down or power off a computer is to remove power from a computer's main components in a controlled way. After a computer is shut down, main components such as CPUs, RAM modules and hard disk drives are powered down, although some internal components, such as an internal clock, may retain power.

1 Windows

In Microsoft Windows, a PC or server is shut down by selecting the Shutdown item from the Start menu on the desktop. Options include shutting down the system and powering off, automatically restarting the system after shutting down, or putting the system into stand-by mode. There is also a shutdown command that can be executed within a command shell window. shutdown.exe is the command-line shutdown application that can shut down the user's computer or another computer on the user's network.

Just like other operating systems, Windows has the option to prohibit selected users from shutting down a computer. On a home PC, every user may have the shutdown option, but in computers on large networks (such as Active Directory), an administrator can revoke the access rights of selected users to shut down a Windows computer. Nowadays there are many software utilities which can automate the task of shutting down a Windows computer, enabling automatic computer control. The Windows Shutdown website lists various software utilities to automate the task of shutting down.

In Windows, a program can shutdown the system by calling the NtShutdownSystem function.[1]

2 Mac OS X

In Mac OS X the computer can be shut down by choosing "Shut Down..." from the Apple Menu or by pressing the power key to bring up the power management dialog box. An administrator may also use the Unix shutdown command as well.

3 Unix and Linux

In Unix and Linux, the shutdown command can be used to turn off or reboot a computer. Only the superuser can shut the system down.

One commonly issued form of this command is shutdown -h now, which will shut down a system immediately. Another one is shutdown -r now to reboot. Another form allows the user to specify an exact time or a delay before shutdown: shutdown -h 20:00 will turn the computer off at 8:00 PM, and shutdown -r -t 60 will automatically reboot the machine within 60 seconds (one minute) of issuing the command."

3.1.2 Access Control

This paragraph shall present an overview of the access and security features of the software that are visible to the user. The following items shall be included, as applicable:

- *How and from whom to obtain a password* (on page 91)
- *How to add, delete, or change passwords under user control* (on page 91)
- *Security and privacy considerations pertaining to the storage and marking of output reports and other media that the user will generate* (on page 91)

3.1.2.1 How and from whom to obtain a password

3.1.2.2 How to add, delete, or change passwords under user control

3.1.2.3 Security and privacy considerations pertaining to the storage and marking of output reports and other media that the user will generate

3.1.3 Installation and Setup

This paragraph shall describe any procedures that the user must perform to be identified or authorized to access or install software on the equipment, to perform the installation, to configure the software, to delete or overwrite former files or data, and to enter parameters for software operation.

- *Procedures to be identified or authorized to access or install software on the equipment* (on page 92)
- *Procedures to perform the installation* (on page 92)
- *Procedures to configure the software* (on page 92)
- *Procedures to delete or overwrite former files or data* (on page 92)
- *Procedures to enter parameters for software operation* (on page 92)

3.1.3.1 Procedures to be identified or authorized to access or install software on the equipment

3.1.3.2 Procedures to perform the installation

3.1.3.3 Procedures to configure the software

3.1.3.4 Procedures to delete or overwrite former files or data

3.1.3.5 Procedures to enter parameters for software operation

3.2 Initiating a Session

This paragraph shall provide step-by-step procedures for beginning work, including any options available. A checklist for problem determination shall be included in case difficulties are encountered.

- *Step-by-step procedures for beginning work* (on page 92)
- *Options available* (on page 92)
- *Checklist for problem determination* (on page 92)

3.2.1 Step-by-step procedures for beginning work

3.2.2 Options available

3.2.3 Checklist for problem determination

3.3 Stopping and Suspending Work

This paragraph shall describe how the user can cease or interrupt use of the software and how to determine whether normal termination or cessation has occurred.

- *How the user can cease or interrupt use of the software* (on page 93)
- *How to determine whether normal termination or cessation has occurred* (on page 93)

3.3.1 How the user can cease or interrupt use of the software

3.3.2 How to determine whether normal termination or cessation has occurred

Draft

Draft

4 PROCESSING REFERENCE GUIDE

This section shall provide the user with procedures for using the software. If procedures are complicated or extensive, additional Sections 6, 7, ... may be added in the same paragraph structure as this section and with titles meaningful to the sections selected. The organization of the document will depend on the characteristics of the software being documented. For example, one approach is to base the sections on the organizations in which users work, their assigned positions, their work sites, or the tasks they must perform. For other software, it may be more appropriate to have Section 5 be a guide to menus, Section 6 be a guide to the command language used, and Section 7 be a guide to functions. Detailed procedures are intended to be presented in subparagraphs of paragraph 5.3. Depending on the design of the software, the subparagraphs might be organized on a function-by-function, menu-b\y-menu, transaction-by-transaction, or other basis. Safety precautions, marked by WARNING or CAUTION, shall be included where applicable

4.1 Capabilities

This paragraph shall briefly describe the interrelationships of the transactions, menus, functions, or other processes in order to provide an overview of the use of the software.

4.2 Conventions

This paragraph shall describe any conventions used by the software, such as the use of colors in displays, the use of audible alarms, the use of abbreviated vocabulary, and the use of rules for assigning names or codes.

4.3 Processing Procedures

This paragraph shall explain the organization of subsequent paragraphs, e.g., by function, by menu, by screen. Any necessary order in which procedures must be accomplished shall be described.

4.3.1 (Aspect of software use)

The title of this paragraph shall identify the function, menu, transaction, or other process being described. This paragraph shall describe and give options and examples, as applicable, of menus, graphical icons, data entry forms, user inputs, inputs from other software or hardware that may affect the software's interface with the user, outputs, diagnostic or error messages or alarms, and help facilities that can provide on-line descriptive or tutorial information. The format for presenting this information can be adapted to the particular characteristics of the software, but a consistent style of presentation shall be used, i.e., the descriptions of menus shall be consistent, the descriptions of transactions shall be consistent among themselves.

4.4 Related Processing

This paragraph shall identify and describe any related batch, offline, or background processing performed by the software that is not invoked directly by the user and is not described in paragraph 5.3. Any user responsibilities to support this processing shall be specified.

4.5 Data Backup

This paragraph shall describe procedures for creating and retaining backup data that can be used to replace primary copies of data in event of errors, defects, malfunctions, or accidents.

4.6 Recovery from Errors, Malfunctions, and Emergencies

This paragraph shall present detailed procedures for restart or recovery from errors or malfunctions occurring during processing and for ensuring continuity of operations in the event of emergencies.

4.7 Messages

This paragraph shall list, or refer to an appendix that lists, all error messages, diagnostic messages, and information messages that can occur while accomplishing any of the user's functions. The meaning of each message and the action that should be taken after each such message shall be identified and described.

4.8 Quick-Reference Guide

If appropriate to the software, this paragraph shall provide or reference a quick-reference card or page for using the software. This quick-reference guide shall summarize, as applicable, frequently-used function keys, control sequences, formats, commands, or other aspects of software use.

Draft

Draft

5 NOTES

This section shall contain any general information that aids in understanding this document (e.g., background information, glossary, rationale). This section shall include an alphabetical listing of all acronyms, abbreviations, and their meanings as used in this document and a list of terms and definitions needed to understand this document. If section 5 has been expanded into section(s) 6, . . ., this section shall be numbered as the next section following section n.

5.1 Use Case(s)

From Wikipedia, the free encyclopedia

"In software and systems engineering, a use case is a list of steps, typically defining interactions between a role (known in Unified Modeling Language (UML) as an "actor") and a system, to achieve a goal. The actor can be a human, an external system, or time.

In systems engineering, use cases are used at a higher level than within software engineering, often representing missions or stakeholder goals. The detailed requirements may then be captured in Systems Modeling Language (SysML) or as contractual statements.

As an important requirement technique, use cases have been widely used in modern software engineering over the last two decades. Use case driven development is a key characteristic of process models and frameworks like Unified Process (UP), Rational Unified Process (RUP), Oracle Unified Method (OUM), etc. With its iterative and evolutionary nature, use case is also a good fit for agile development."

The use case(s) describe the expected activities associated with various functional roles:

- 1 Operating System Administration** (on page 101)
- 2 Python Language Administration** (on page 107)
- 3 "tsWxGTUI_PyVx" Toolkit Development** (see "'tsWxGTUI" Toolkit Development" on page 107)
- 4 Python Application Development** (on page 108)
- 5 Python Application Usage** (on page 108)
- 6 System Troubleshooting & Maintenance** (on page 108)

5.1.1 Booting

Excerpted From Wikipedia, the free encyclopedia

"In computing, booting (also known as booting up) is the initial set of operations that a computer system performs after electrical power to the CPU is switched on or when the computer is reset. The process begins when a computer is turned on for the first time, is re-energized after being turned off, when it is reset or when the operator invokes a LOAD[NB 1] function from the console, and ends when the computer is ready to perform its normal operations. On modern general purpose computers, this can take tens of seconds and typically involves performing a power-on self-test, locating and initializing peripheral devices, and then finding, loading and starting an operating system. Many computer systems also allow these operations to be initiated by a software command without cycling power, in what is known as a soft reboot, though some of the initial operations might be skipped on a soft reboot. A boot loader is a computer program that loads the main operating system or runtime environment for the computer after completion of the self-tests.

The computer term boot is short for bootstrap[1][2] or bootstrap load and derives from the phrase to pull oneself up by one's bootstraps.[3] The usage calls attention to the requirement that, if most software is loaded onto a computer by other software already running on the computer, some mechanism must exist to load the initial software onto the computer.[4] Early computers used a variety of ad-hoc methods to get a small program into memory to solve this problem. The invention of read-only memory (ROM) of various types solved this paradox by allowing computers to be shipped with a start up program that could not be erased. Growth in the capacity of ROM has allowed ever more elaborate start up procedures to be implemented.

On general purpose computers, the boot process begins with the execution of an initial program stored in boot ROMs or read in another fashion. In some older computers, the initial program might have been the application to run, if no operating system was used, or the operating system. In other computers, the initial program is a boot loader that may then load into random-access memory (RAM), from nonvolatile secondary storage (such as a hard disk drive) or, in some older computers, from a medium such as punched cards, punched tape, or magnetic tape, the binary code of an operating system or runtime environment and then execute it. If the boot loader is limited in its size and capabilities, it may, instead, load a larger and more capable secondary boot loader, which would then load the operating system or runtime environment. Some embedded systems do not require a noticeable boot sequence to begin functioning and when turned on may simply run operational programs that are stored in ROM."

5.1.2 Shutdown

Excerpted From Wikipedia, the free encyclopedia:

"To shut down or power off a computer is to remove power from a computer's main components in a controlled way. After a computer is shut down, main components such as CPUs, RAM modules and hard disk drives are powered down, although some internal components, such as an internal clock, may retain power.

Implementations:

1 Windows

In Microsoft Windows, a PC or server is shut down by selecting the Shutdown item from the Start menu on the desktop. Options include shutting down the system and powering off, automatically restarting the system after shutting down, or putting the system into stand-by mode. There is also a shutdown command that can be executed within a command shell window. shutdown.exe is the command-line shutdown application that can shut down the user's computer or another computer on the user's network.

Just like other operating systems, Windows has the option to prohibit selected users from shutting down a computer. On a home PC, every user may have the shutdown option, but in computers on large networks (such as Active Directory), an administrator can revoke the access rights of selected users to shut down a Windows computer. Nowadays there are many software utilities which can automate the task of shutting down a Windows computer, enabling automatic computer control. The Windows Shutdown website lists various software utilities to automate the task of shutting down.

In Windows, a program can shutdown the system by calling the NtShutdownSystem function.[1]

2 Mac OS X

In Mac OS X the computer can be shut down by choosing "Shut Down..." from the Apple Menu or by pressing the power key to bring up the power management dialog box. An administrator may also use the Unix shutdown command as well.

3 Unix and Linux

In Unix and Linux, the shutdown command can be used to turn off or reboot a computer. Only the superuser can shut the system down.

One commonly issued form of this command is shutdown -h now, which will shut down a system immediately. Another one is shutdown -r now to reboot. Another form allows the user to specify an exact time or a delay before shutdown: shutdown -h 20:00 will turn the computer off at 8:00 PM, and shutdown -r -t 60 will automatically reboot the machine within 60 seconds (one minute) of issuing the command."

5.1.3 Operating System Administration

- 1 **System Administrator** (on page 109) - In this role, one or more individuals specify, plan, supervise and/or perform the installation, configuration, maintenance, repair and support of the computer hardware, operating system, application software and network to be used by Software Engineers and System Operators.
- 2 **Field Service Personnel** (on page 112) - In this role, one or more individuals install, repair and replace a manufacturer's hardware and software, at customer locations. They use diagnostic and maintenance programs, with associated command line or graphical user interfaces, to configure, test, tune and troubleshoot various hardware and software issues.

5.1.3.1 Apple Computer, Inc.

5.1.3.1.1 Apple Mac OS X Unix-like, Darwin-based Use Case

Operation of a typical computer system advances through various stages:

- 1 Manual Power-ON of Host Computer Hardware (**Booting** (on page 100))
- 2 Automatic reset of Host Computer Hardware

- 3 Automatic startup of Host Computer Operating System Software
- 4 Automatic startup of Host Computer Pixel-mode Graphical User Interface Software (**login**)
- 5 Optional installation of new or updated Host Computer Operating System Libraries and Software
- 6 Optional installation of new or updated "tsWxGTUI_PyVx" Toolkit Application Libraries and Software
- 7 Manual startup of Host Computer Command Line Interface Software (**Applications -> iTerm** or **Applications -> Utilities -> terminal**)
- 8 Manual startup of "tsWxGTUI_PyVx" Toolkit Application Software (**python3.2 test_tsWxWidgets.py**)
 - a) Automatic startup of Character-mode Graphical User Interface Software
 - b) Automatic startup of Character-mode Top Level GUI objects
 - c) Automatic startup of Character-mode Lower Level GUI objects
 - d) Automatic shutdown of Character-mode Lower Level GUI objects
 - e) Automatic shutdown of Character-mode Top Level GUI objects
 - f) Automatic shutdown of Character-mode Graphical User Interface Software (**ctrl-c**)
- 9 Manual shutdown of Host Computer Command Line Interface Software (**exit**)
- 10 Manual shutdown of Host Computer Pixel-mode Graphical User Interface Software (**logout**)
- 11 Manual shutdown of Host Computer Operating System Software (**shutdown**)
- 12 Manual Power-OFF of Host Computer Hardware (**Shutdown** (on page 100))

5.1.3.2 FreeBSD Foundation

5.1.3.2.1 PC-BSD (FreeBSD) Unix-like Use Case

Operation of a typical computer system advances through various stages:

- 1 Manual Power-ON of Host Computer Hardware (**Booting** (on page 100))
- 2 Automatic reset of Host Computer Hardware
- 3 Automatic startup of Host Computer Operating System Software
- 4 Automatic startup of Host Computer Pixel-mode Graphical User Interface Software (**login**)
- 5 Optional installation of new or updated Host Computer Operating System Libraries and Software
- 6 Optional installation of new or updated "tsWxGTUI" Toolkit Application Libraries and Software
- 7 Manual startup of Host Computer Command Line Interface Software (**Applications -> Accessories -> terminal**)
- 8 Manual startup of "tsWxGTUI" Toolkit Application Software (**python3.2 test_tsWxWidgets.py**)
 - a) Automatic startup of Character-mode Graphical User Interface Software
 - b) Automatic startup of Character-mode Top Level GUI objects
 - c) Automatic startup of Character-mode Lower Level GUI objects
 - d) Automatic shutdown of Character-mode Lower Level GUI objects

- e) Automatic shutdown of Character-mode Top Level GUI objects
- f) Automatic shutdown of Character-mode Graphical User Interface Software (**ctrl-c**)
- 9** Manual shutdown of Host Computer Command Line Interface Software (**exit**)
- 10** Manual shutdown of Host Computer Pixel-mode Graphical User Interface Software (**System -> Logout**)
- 11** Manual shutdown of Host Computer Operating System Software (**System -> Shutdown**)
- 12** Manual Power-OFF of Host Computer Hardware (**Shutdown** (on page 100))

5.1.3.3 Illumos Foundation

5.1.3.3.1 OpenIndiana (OpenSolaris) Unix-like Use Case

Operation of a typical computer system advances through various stages:

- 1** Manual Power-ON of Host Computer Hardware (**Booting** (on page 100))
- 2** Automatic reset of Host Computer Hardware
- 3** Automatic startup of Host Computer Operating System Software
- 4** Automatic startup of Host Computer Pixel-mode Graphical User Interface Software (**login**)
- 5** Optional installation of new or updated Host Computer Operating System Libraries and Software
- 6** Optional installation of new or updated "tsWxGTUI" Toolkit Application Libraries and Software
- 7** Manual startup of Host Computer Command Line Interface Software (**Applications -> Accessories -> terminal**)
- 8** Manual startup of "tsWxGTUI" Toolkit Application Software (**python3.2 test_tsWxWidgets.py**)
 - a) Automatic startup of Character-mode Graphical User Interface Software
 - b) Automatic startup of Character-mode Top Level GUI objects
 - c) Automatic startup of Character-mode Lower Level GUI objects
 - d) Automatic shutdown of Character-mode Lower Level GUI objects
 - e) Automatic shutdown of Character-mode Top Level GUI objects
 - f) Automatic shutdown of Character-mode Graphical User Interface Software (**ctrl-c**)
- 9** Manual shutdown of Host Computer Command Line Interface Software (**exit**)
- 10** Manual shutdown of Host Computer Pixel-mode Graphical User Interface Software (**System -> Logout**)
- 11** Manual shutdown of Host Computer Operating System Software (**System -> Shutdown**)
- 12** Manual Power-OFF of Host Computer Hardware (**Shutdown** (on page 100))

5.1.3.4 Microsoft Corporation

5.1.3.4.1 Microsoft Windows Use Case

NOTES:

1) Once the "tsWxGTUI" Toolkit has been installed on Microsoft Windows, its Command Line Interface Software is available. However, its Graphical-style User Interface Software is NOT available without the installation of "Cygwin", the free, Linux-like Command Line Interface and GNU Tools from Red Hat.

2) Do NOT use this procedure after the installation of "Cygwin". Instead, use the procedure in *Microsoft Windows & Cygwin (Red Hat) Use Case* (on page 104).

Operation of a typical computer system advances through various stages:

- 1 Manual Power-ON of Host Computer Hardware (**Booting** (on page 100))
- 2 Automatic reset of Host Computer Hardware
- 3 Automatic startup of Host Computer Operating System Software
- 4 Automatic startup of Host Computer Pixel-mode Graphical User Interface Software (**login**)
- 5 Optional installation of new or updated Host Computer Operating System Libraries and Software
- 6 Optional installation of new or updated "tsWxGTUI" Toolkit Application Libraries and Software
- 7 Manual startup of Host Computer Command Line Interface Software (**Start -> Programs -> Accessories -> Command Prompt**)
- 8 Manual startup of "tsWxGTUI" Toolkit Command Line Interface Software Application Software (**python tsLinesOfCodeProjectMetrics.py**)
- 9 Manual shutdown of Host Computer Command Line Interface Software (**exit**)
- 10 Manual shutdown of Host Computer Pixel-mode Graphical User Interface Software (**logout**)
- 11 Manual shutdown of Host Computer Operating System Software (**shutdown**)
- 12 Manual Power-OFF of Host Computer Hardware (**Shutdown** (on page 100))

5.1.3.4.2 Microsoft Windows & Cygwin (Red Hat) Use Case

NOTES:

1) Once the "tsWxGTUI_PyVx" Toolkit has been installed on Microsoft Windows, its Command Line Interface Software is available. However, its Graphical-style User Interface Software is NOT available without the installation of "Cygwin", the free, Linux-like Command Line Interface and GNU Tools from Red Hat.

2) Use this procedure only AFTER the installation of "Cygwin". Until then, use the procedure in *Microsoft Windows Use Case* (on page 104).

Operation of a typical computer system advances through various stages:

- 1 Manual Power-ON of Host Computer Hardware (**Booting** (on page 100))

- 2 Automatic reset of Host Computer Hardware
- 3 Automatic startup of Host Computer Operating System Software
- 4 Automatic startup of Host Computer Pixel-mode Graphical User Interface Software (**login**)
- 5 Optional installation of new or updated Host Computer Operating System Libraries and Software
- 6 Optional installation of new or updated "tsWxGTUI_PyVx" Toolkit Application Libraries and Software
- 7 Manual startup of Host Computer Command Line Interface Software (**Start -> Programs -> Cygwin -> Cygwin Bash Shell** or **Start -> Programs -> Cygwin-X -> XWin Server**)
- 8 Manual startup of "tsWxGTUI_PyVx" Toolkit Application Software (**python test_tsWxWidgets.py**)
 - a) Automatic startup of Character-mode Graphical User Interface Software
 - b) Automatic startup of Character-mode Top Level GUI objects
 - c) Automatic startup of Character-mode Lower Level GUI objects
 - d) Automatic shutdown of Character-mode Lower Level GUI objects
 - e) Automatic shutdown of Character-mode Top Level GUI objects
 - f) Automatic shutdown of Character-mode Graphical User Interface Software (**ctrl-c**)
- 9 Manual shutdown of Host Computer Command Line Interface Software (**exit**)
- 10 Manual shutdown of Host Computer Pixel-mode Graphical User Interface Software (**logout**)
- 11 Manual shutdown of Host Computer Operating System Software (**shutdown**)
- 12 Manual Power-OFF of Host Computer Hardware (**Shutdown** (on page 100))

5.1.3.5 Red Hat, Inc.

5.1.3.5.1 Fedora (Red Hat) Linux Use Case

Operation of a typical computer system advances through various stages:

- 1 Manual Power-ON of Host Computer Hardware (**Booting** (on page 100))
- 2 Automatic reset of Host Computer Hardware
- 3 Automatic startup of Host Computer Operating System Software
- 4 Automatic startup of Host Computer Pixel-mode Graphical User Interface Software (**login**)
- 5 Optional installation of new or updated Host Computer Operating System Libraries and Software
- 6 Optional installation of new or updated "tsWxGTUI" Toolkit Application Libraries and Software
- 7 Manual startup of Host Computer Command Line Interface Software (**Applications -> Accessories -> terminal**)
- 8 Manual startup of "tsWxGTUI" Toolkit Application Software (**python3.2 test_tsWxWidgets.py**)
 - a) Automatic startup of Character-mode Graphical User Interface Software
 - b) Automatic startup of Character-mode Top Level GUI objects
 - c) Automatic startup of Character-mode Lower Level GUI objects

- d) Automatic shutdown of Character-mode Lower Level GUI objects
- e) Automatic shutdown of Character-mode Top Level GUI objects
- f) Automatic shutdown of Character-mode Graphical User Interface Software (**ctrl-c**)
- 9** Manual shutdown of Host Computer Command Line Interface Software (**exit**)
- 10** Manual shutdown of Host Computer Pixel-mode Graphical User Interface Software (**System -> Logout**)
- 11** Manual shutdown of Host Computer Operating System Software (**System -> Shutdown**)
- 12** Manual Power-OFF of Host Computer Hardware (**Shutdown** (on page 100))

5.1.3.5.2 Scientific (CentOS Red Hat Enterprise) Linux Use Case

Operation of a typical computer system advances through various stages:

- 1** Manual Power-ON of Host Computer Hardware (**Booting** (on page 100))
- 2** Automatic reset of Host Computer Hardware
- 3** Automatic startup of Host Computer Operating System Software
- 4** Automatic startup of Host Computer Pixel-mode Graphical User Interface Software (**login**)
- 5** Optional installation of new or updated Host Computer Operating System Libraries and Software
- 6** Optional installation of new or updated "tsWxGTUI" Toolkit Application Libraries and Software
- 7** Manual startup of Host Computer Command Line Interface Software (**Applications -> Accessories -> terminal**)
- 8** Manual startup of "tsWxGTUI" Toolkit Application Software (**python3.2 test_tsWxWidgets.py**)
 - a) Automatic startup of Character-mode Graphical User Interface Software
 - b) Automatic startup of Character-mode Top Level GUI objects
 - c) Automatic startup of Character-mode Lower Level GUI objects
 - d) Automatic shutdown of Character-mode Lower Level GUI objects
 - e) Automatic shutdown of Character-mode Top Level GUI objects
 - f) Automatic shutdown of Character-mode Graphical User Interface Software (**ctrl-c**)
- 9** Manual shutdown of Host Computer Command Line Interface Software (**exit**)
- 10** Manual shutdown of Host Computer Pixel-mode Graphical User Interface Software (**System -> Logout**)
- 11** Manual shutdown of Host Computer Operating System Software (**System -> Shutdown**)
- 12** Manual Power-OFF of Host Computer Hardware (**Shutdown** (on page 100))

5.1.3.6 Ubuntu Foundation

5.1.3.6.1 Ubuntu Linux Use Case

Operation of a typical computer system advances through various stages:

- 1** Manual Power-ON of Host Computer Hardware (**Booting** (on page 100))

- 2 Automatic reset of Host Computer Hardware
- 3 Automatic startup of Host Computer Operating System Software
- 4 Automatic startup of Host Computer Pixel-mode Graphical User Interface Software (**login**)
- 5 Optional installation of new or updated Host Computer Operating System Libraries and Software
- 6 Optional installation of new or updated "tsWxGTUI_PyVx" Toolkit Application Libraries and Software
- 7 Manual startup of Host Computer Command Line Interface Software (**Applications -> Accessories -> terminal**)
- 8 Manual startup of "tsWxGTUI_PyVx" Toolkit Application Software (**python3.2 test_tsWxWidgets.py**)
 - a) Automatic startup of Character-mode Graphical User Interface Software
 - b) Automatic startup of Character-mode Top Level GUI objects
 - c) Automatic startup of Character-mode Lower Level GUI objects
 - d) Automatic shutdown of Character-mode Lower Level GUI objects
 - e) Automatic shutdown of Character-mode Top Level GUI objects
 - f) Automatic shutdown of Character-mode Graphical User Interface Software (**ctrl-c**)
- 9 Manual shutdown of Host Computer Command Line Interface Software (**exit**)
- 10 Manual shutdown of Host Computer Pixel-mode Graphical User Interface Software (**System -> Logout**)
- 11 Manual shutdown of Host Computer Operating System Software (**System -> Shutdown**)
- 12 Manual Power-OFF of Host Computer Hardware (**Shutdown** (on page 100))

5.1.4 "tsWxGTUI" Toolkit Development

- 1 **Software Engineer** (on page 110) - In this role, one or more individuals specify, plan, supervise and/or perform the design, development, coding, testing, debugging, maintenance and support of application programs, with command line or graphical user interface, to be used by System Operators.
- 2 **System Operator** (on page 111) - In this role, one or more individuals use various application programs, with associated command line or graphical user interfaces, to interactively supervise communication, control, data base, diagnostic, instrumentation or simulation activities.

5.1.5 Python Language Administration

- 1 **System Administrator** (on page 109) - In this role, one or more individuals specify, plan, supervise and/or perform the installation, configuration, maintenance, repair and support of the computer hardware, operating system, application software and network to be used by Software Engineers and System Operators.

5.1.5.1 Python Software Foundation

5.1.6 "tsWxGTUI" Toolkit Development

- 1 **Software Engineer** (on page 110) - In this role, one or more individuals specify, plan, supervise and/or perform the design, development, coding, testing, debugging, maintenance and support of application programs, with command line or graphical user interface, to be used by System Operators.
- 2 **System Operator** (on page 111) - In this role, one or more individuals use various application programs, with associated command line or graphical user interfaces, to interactively supervise communication, control, data base, diagnostic, instrumentation or simulation activities.

5.1.7 Python Application Development

- 1 **Software Engineer** (on page 110) - In this role, one or more individuals specify, plan, supervise and/or perform the design, development, coding, testing, debugging, maintenance and support of application programs, with command line or graphical user interface, to be used by System Operators.
- 2 **System Operator** (on page 111) - In this role, one or more individuals use various application programs, with associated command line or graphical user interfaces, to interactively supervise communication, control, data base, diagnostic, instrumentation or simulation activities.

5.1.8 Python Application Usage

- 1 **Software Engineer** (on page 110) - In this role, one or more individuals specify, plan, supervise and/or perform the design, development, coding, testing, debugging, maintenance and support of application programs, with command line or graphical user interface, to be used by System Operators.
- 2 **System Operator** (on page 111) - In this role, one or more individuals use various application programs, with associated command line or graphical user interfaces, to interactively supervise communication, control, data base, diagnostic, instrumentation or simulation activities.

5.1.9 System Troubleshooting & Maintenance

- 1 **System Administrator** (on page 109) - In this role, one or more individuals specify, plan, supervise and/or perform the installation, configuration, maintenance, repair and support of the computer hardware, operating system, application software and network to be used by Software Engineers and System Operators.
- 2 **Software Engineer** (on page 110) - In this role, one or more individuals specify, plan, supervise and/or perform the design, development, coding, testing, debugging, maintenance and support of application programs, with command line or graphical user interface, to be used by System Operators.
- 3 **System Operator** (on page 111) - In this role, one or more individuals use various application programs, with associated command line or graphical user interfaces, to interactively supervise communication, control, data base, diagnostic, instrumentation or simulation activities.

- 4 Field Service Personnel** (on page 112) - In this role, one or more individuals install, repair and replace a manufacturer's hardware and software, at customer locations. They use diagnostic and maintenance programs, with associated command line or graphical user interfaces, to configure, test, tune and troubleshoot various hardware and software issues.

5.1.10 System Administrator

Based on: http://en.wikipedia.org/wiki/System_administrator

A System Administrator is typically responsible for supervising and/or performing the following:

- Installation, configuration, maintenance, repair and support of all system hardware, operating system and application software and network components.
- Scripting or light programming.
- Project management for systems-related projects.
- Supervising or training computer operators.
- Consultant for computer problems beyond the knowledge of technical support staff.
- To perform his or her job well, a System Administrator must demonstrate a blend of technical skills and responsibility.

The subject matter of System Administration includes computer systems and the ways people use them in an organization. This entails a knowledge of operating systems and applications, as well as hardware and software troubleshooting, but also knowledge of the purposes for which people in the organization use the computers. Perhaps the most important skill for a System Administrator is problem solving -- frequently under various sorts of constraints and stress. The System Administrator is on call when a computer system goes down or malfunctions, and must be able to quickly and correctly diagnose what is wrong and how best to fix it. System Administrators are not Software Engineers. It is not usually within their duties to design or write new application software. However, System Administrators must understand the behavior of software in order to deploy it and to troubleshoot problems, and generally know several programming languages used for scripting or automation of routine tasks.

5.1.10.1 Enabling Local Host Login

5.1.10.2 Enabling Remote Host Login

5.1.11 Software Engineer

Derived from: <http://www.bls.gov/oco/ocos303.htm>

Computer Software Engineers design and develop software for use by System Operators. They perform the following:

- Begin by analyzing users' needs and then design, test, and develop software to meet those needs.
- Apply the theories and principles of computer science and mathematical analysis to create, test, and evaluate the systems, application software and command line or graphical user interface that make computers work.
- During this process they create flowcharts, diagrams, and other documentation, and may also create the detailed sets of instructions, called algorithms, that actually tell the computer what to do.
- They may also be responsible for converting these instructions into a computer language, a process called programming or coding, but this usually is the responsibility of computer programmers.

5.1.11.1 Planner

5.1.11.1.1 Analyst

5.1.11.1.2 Architect

5.1.11.2 Implementor

5.1.11.2.1 Designer

5.1.11.2.2 Documentor

5.1.11.2.3 Coder

5.1.11.2.4 Integrator

5.1.11.2.5 Maintainer

5.1.11.2.6 Troubleshooter

5.1.11.3 Qualifier

5.1.11.3.1 Tester

5.1.11.3.2 Validator

5.1.11.3.3 Acceptor

5.1.11.4 Producer

5.1.11.4.1 Release Packager

5.1.11.4.2 Release Distributor

5.1.12 System Operator

Derived from <http://whatis.techtarget.com/definition/system-operator-sysop>

A System Operator is the person who runs the day-to-day operation of the computer system. The term suggests a person who is available when the system is.

The System Operator is ultimately the end-user of application programs, with associated command line or graphical user interfaces, that interactively supervise various communication, control, data base, diagnostic, instrumentation or simulation activities.

At earlier stages in the computer system's installation, configuration and software development process, the System Operator may temporarily be a **System Administrator** (on page 109) or **Software Engineer** (on page 110).

At later stages in the computer system's maintenance, the System Operator may temporarily be one of the **Field Service Personnel** (on page 112).

The activities are somewhat role-specific and system specific. The following use cases illustrate the similarities and differences:

- *Apple Mac OS X Use Case* (see "*Apple Mac OS X Unix-like, Darwin-based Use Case*" on page 101)
- *Microsoft Windows & Cygwin Use Case* (see "*Microsoft Windows & Cygwin (Red Hat) Use Case*" on page 104)
- *Ubuntu Linux Use Case* (on page 106)
- FreeBSD/PC-BSD Unix Use Case

5.1.12.1 Local Host Login

5.1.12.2 Remote Host Login

5.1.12.3 Configuring Local Host Shell

- 1 Size
- 2 Position
- 3 Font
- 4 Terminal Emulator

5.1.12.4 Launch Command Line Interface Application

5.1.12.5 Launch Graphical-style User Interface Application

5.1.12.6 Terminating Command Line Interface Application

5.1.12.7 Terminating Graphical-style User Interface Application

5.1.13 Field Service Personnel

The Field Service Personnel includes those engineers and technicians who:

- 1 Install a manufacturer's hardware and software at a customer location.
- 2 Use diagnostic and maintenance programs, with associated command line or graphical user interfaces, to interactively configure, test, tune and troubleshoot the previously installed hardware and software.
- 3 Repair or replace defective hardware and software.

5.2 Baseline Toolkit Development Platforms

Two of the following hardware and software platforms were already available, in 2007, when planning and prototyping began for the "tsWxGTUI" Toolkit development project.

Limited financial resources dictated that the same hardware be used to plan, develop, test, document, manage software configurations and versions as well as backup the project's tools, data and work in-progress.

- ***Apple 27" iMac Desktop*** (on page 114) - 2013-model year 3.5 GHz Intel "Core i7" processor, with four independent processor "cores" with sufficient performance, resources and expansion capabilities to serve as premium-level baseline development and operator platforms. Its 256KB L2 Cache (per Core), 8 MB L3 Cache, 16.0 GB of 1600 MHz DDR3 SDRAM, 128 GB APPLE SSD, SD0128F Media was used to run Apple's Mac OS X and the hypervisor virtualization applications (Parallels and VMware Fusion) that supported various guest operating systems. Its 3 TB APPLE HDD ST3000DM001 Media was used to store and run configured versions of the Linux, Microsoft Windows and Unix guest operating systems. It was also configured with Apple DVD±RW/CD-RW SuperDrive, NVIDIA GeForce GTX 775M 2 GB VRAM, 27" widescreen 2560 x 1440 TFT active-matrix display, "Apple Wireless Trackpad" and "Logitech TracMan Wheel Trackball".
- ***Apple 17" MacBook Pro Laptop*** (on page 118) - 2007-model year, 2.33 GHz Intel Core 2 Duo processor-based laptop with sufficient performance, resources and expansion capabilities to serve as the high-level baseline development and operator platforms. Its 160 GB internal hard drive was used to run Apple's Mac OS X and the hypervisor virtualization applications (Parallels and VMware Fusion) that supported various guest operating systems. Its 1.5 TB external hard drive was used to store and run configured versions of the Linux, Microsoft Windows and Unix guest operating systems.
- ***Dell 15.6" Inspiron 7000 Laptop*** (on page 123) - 1999-model year, 366 MHz Intel Pentium II-based laptop with marginal performance, resources and expansion capabilities to serve as the low-level baseline development and operator platforms. Its interchangeable 32 GB hard drives were used to run Windows XP or Ubuntu 12.04 Linux. The platform's limited memory and available PCMCIA network adapters were incompatible with later versions of Windows or with other Linux distributions.

5.2.1 Apple 27" iMac Desktop

5.2.1.1 Hardware

Apple 27" iMac Desktop:

- 3.5 GHz Intel "Core i7" processor, with four independent processor "cores"
- 256KB L2 Cache (per Core)
- 8 MB L3 Cache
- 16.0 GB of 1600 MHz DDR3 SDRAM
- 128 GB APPLE SSD SD0128F Media
- 3 TB APPLE HDD ST3000DM001 Media
- Apple DVD±RW/CD-RW SuperDrive
- NVIDIA GeForce GTX 775M 2 GB VRAM
- 27" widescreen 2560 x 1440 TFT active-matrix display.
- "Apple Wireless Trackpad"
- "Logitech TracMan Wheel Trackball"

5.2.1.2 Software (Draft)

5.2.1.2.1 Mac OS X 32-bit/64-bit

Apple Mac OS X (upgraded from 10.8 Mountain Lion to 10.9 Mavericks)

Usability Issue(s):

- The built-in 128GB Solid State Drive is dynamically loaded with Mac OS X and application code as needed to optimize performance.

5.2.1.2.2 Parallels Desktop 9 for Mac

Parallels Desktop for Mac (upgraded from 3 through 9)

Usability Issue(s):

- Highly configurable and user friendly.
- Can share Host resources with Guest OS
- Tools provided only for most popular Guest Operating Systems (Microsoft Windows, Fedora Linux, Scientific Linux, Ubuntu Linux). Tools are NOT provided for PC-BSD / Solaris / OpenIndiana. The tools are distributed as a mountable ISO-image containing a native shell install script and installable files. Once installed, the Tools install updates automatically.
- Tools for MS Windows install and update automatically.

5.2.1.2.2.1 Active Virtual Machines

5.2.1.2.2.1.1 *Cygwin (Unix-like Environment) for MS Windows*

Cygwin Unix-like Environment for MS Windows (upgraded from 1.6 to 1.8)

Usability Issue(s):

- Supports Command Line Interface with bash shell and optional xterm session.
- Optional pixel-mode Graphical-style User Interface requires X11.
- Optional character-mode Graphical-style User Interface requires nCurses terminal interface library.
- Somewhat behind in latest Python 2.x version (2.7.3) and cannot be updated via Cygwin setup nor via Python download.
- Somewhat behind in latest Python 3.x version (3.2.3) and cannot be updated via Cygwin setup nor via Python download.
- Somewhat behind in latest GNU utility versions

5.2.1.2.2.1.2 *Linux Ubuntu 12.04 LTS 32-bit*

Ubuntu Linux Virtual Machine (upgraded from 10.04 to 12.04 LTS)

Usability Issue(s):

- Easiest Linux to download, install, configure, update and upgrade.
- Offers UNITY (preferred), GNOME and KDE desktops.
- Marketing Claim: "Ubuntu is easy to use. And it comes with thousands of free applications. Ubuntu does everything you need it to. It'll work with your existing PC files, printers, cameras and MP3 players."
- Easy installation, configuration, enhancement and upgrade.
- Choice of UNITY, GNOME, KDE desktops, etc.
- Extensive library of optional packages available from primary (Ubuntu) and third party sources.
- Competitive and compatible with Microsoft Windows and Apple Mac OS X.
- Able to establish network connection to DropBox to share latest changes to "tsWxGTUI" Toolkit

5.2.1.2.2.1.3 *Linux Fedora 20 64-bit*
5.2.1.2.2.1.4 *Linux Scientific (CENTOS) 6.4 64-bit*
5.2.1.2.2.1.5 *Microsoft Windows 8.1/8 Professional 32-bit*

Microsoft Windows 8 Professional Virtual Machine

Usability Issue(s):

- Only Professional Edition supports Microsoft Office XP Professional
- Only Professional Edition supports Microsoft Visio XP Professional
- Only Professional Edition supports Author-It 5.5.0 (unless Microsoft Jet Database is replaced by Microsoft SQL Server or SQL Express)
- Does NOT support 3D graphics accelerator
- Requires Cygwin Unix-like Environment
- Python 2.x and 3.x Native Windows versions can be installed via Python download but do not support curses
- Able to establish network connection to DropBox to share latest changes to "tsWxGTUI" Toolkit

5.2.1.2.2.1.6 *Microsoft Windows 7 Professional 32-bit*

Microsoft Windows 7 Professional Virtual Machine

Usability Issue(s):

- Only Professional Edition supports Microsoft Office XP Professional
- Only Professional Edition supports Microsoft Visio XP Professional
- Only Professional Edition supports Author-It 5.5.0 (with Microsoft Jet Database)
- Does NOT support 3D graphics accelerator
- Requires Cygwin Unix-like Environment
- Python 2.x and 3.x Native Windows versions can be installed via Python download but do not support curses
- Able to establish network connection to DropBox to share latest changes to "tsWxGTUI" Toolkit

5.2.1.2.2.1.7 *Microsoft Windows XP 32-bit (Deprecated)*

Microsoft Windows XP Professional (upgraded to SPK3)

Usability Issue(s):

- End-of-Life scheduled for 04/20/2014
- Requires Cygwin Unix-like Environment
- Supports Microsoft Office XP Professional (Deprecated)
- Supports Microsoft Visio XP Professional (Deprecated)
- Python 2.x and 3.x Native Windows versions can be installed via Python download but do not support curses
- Able to establish network connection to DropBox to share latest changes to "tsWxGTUI" Toolkit

5.2.1.2.2.1.8 Unix Open Indiana 151a8 (Solaris 11 / SunOS 5.11) 32-bit

OpenSolaris Unix Virtual Machine (Release 11)

Usability Issue(s):

- Parallels Tools NOT available
- Built-in network connections
- Oracle/Sun no longer support this or newer versions
- Unable to download and install DropBox to share latest changes to "tsWxGTUI" Toolkit

5.2.1.2.2.1.9 Unix PC-BSD 9.2 64-bit

FreeBSD-based Unix Virtual Machine (Release 9.2) with built-in GUI Desktop.

Usability Issue(s):

- Parallels Tools NOT available
- Built-in network connections
- Unable to download and install DropBox to share latest changes to "tsWxGTUI" Toolkit

5.2.2 Apple 17" MacBook Pro Laptop

5.2.2.1 Hardware

Apple 17" MacBook Pro Laptop:

- 2.33 GHz "Core 2 Duo" processor (T7600), with two independent processor "cores"
- 4 MB shared "on chip" level 2 cache
- 667 MHz frontside bus with 2.0 GB of 667 MHz PC2-5300 DDR2 SDRAM
- 160 GB hard drive with "Sudden Motion Sensor" technology
- 8X dual-layer DVD±RW/CD-RW SuperDrive
- ATI Mobility Radeon X1600 graphics processor with 256 MB of GDDR3 video memory and dual-link DVI functionality
- 17" widescreen 1680x1050 TFT active-matrix display (glossy display option available).
- fiber-optic-based "ambient light sensor" that adjusts keyboard illumination and screen brightness
- "scrolling TrackPad"
- "MagSafe" power connector
- integrated iSight video camera
- Apple Remote

Connectivity includes:

- ExpressCard/34 slot
- AirPort Extreme (802.11n as well as 802.11g after a firmware update)
- Bluetooth 2.0+EDR
- Gigabit Ethernet
- FireWire "400" port
- Firewire "800" port
- Three USB 2.0 "high speed" ports
- optical digital audio in/out
- DVI out

Enhancements:

UPGRADE	USAGE	NOTES	PARTITIONS	MAKE	MODEL / COST
Memory	Maximize memory available for Apple Mac OS X and associated applications	Replaced Apple 2GB RAM card with 2GB RAM card and added another 2GB RAM card		Crucial	\$108.00

High Speed External Hard Drive	1.5TB 7200RPM SATA Hard Drive	Early releases required firmware updates/drive replacement to prevent data becoming inaccessible	Seagate	ST31500341AS \$90.00
Bootable High Speed Hard Drive Enclosure/Dock	Supply power, interfaces and mechanical mounting for SATA hard drive	Vulnerable to loss of external electric power and to accidental SATA/USB cable disconnection.	Seagate Drive 1.5TB2 <ul style="list-style-type: none"> ▪ Boot ▪ Archive ▪ GuestOS ▪ Media 	Thermaltake BlacX N0028USU \$59.95
High Speed External DVD Drive	USB Powered, Netbook booting support, Windows 7 & MAC compatible replaces non-functioning internal DVD writer	Vulnerable to loss of external electric power and to accidental USB cable disconnection.	Samsung Super WriteMaster Slim External DVD Writer <ul style="list-style-type: none"> ▪ 8x DVD+/-R write ▪ 8x DVD-ROM read ▪ 24X CD-ROM Read 	Samsung SE-S084D/TSBS
Non-bootable, Backup High Speed Hard Drive Enclosure/Dock	Supply power, interfaces and mechanical mounting for SATA hard drive	Vulnerable to loss of external electric power and to accidental SATA/USB cable disconnection.	Seagate Drive 1.5TB1 <ul style="list-style-type: none"> ▪ Boot ▪ Archive ▪ GuestOS ▪ Media 	Vantec NexStar NST-D200SU \$68.37
Bootable High Speed Hard Drive Adapter	Interface laptop with external SATA hard drive		Other World Computing	OWCEXP34SA TA2P1 \$19.00

Usability Issue(s):

- Graphics hardware will NOT Support MAC OS X Mountain Lion 10.8

5.2.2.2 Software

5.2.2.2.1 Mac OS X 32-bit/64-bit

Apple Mac OS X (upgraded from 10.4 Tiger through 10.7 Lion)

Usability Issue(s):

- Graphics hardware will not support upgrade to Mountain Lion 10.8.

5.2.2.2.2 Parallels Desktop 8 for Mac

Parallels Desktop for Mac (upgraded from 3 through 8)

Usability Issue(s):

- Highly configurable and user friendly.
- Can share Host resources with Guest OS
- Tools provided only for most popular Guest Operating Systems (Microsoft Windows, Ubuntu Linux). Tools are NOT provided for Solaris / OpenIndiana. The tools are distributed as a mountable ISO-image containing a native shell install script and installable files.
- Tools for MS Windows install automatically.

5.2.2.2.2.1 Active Virtual Machines

- *Cygwin (Unix-like Environment) for MS Windows* (on page 115)
- *Microsoft Windows 8.1/8 Professional 32-bit* (on page 116)
- *Microsoft Windows 7 Professional 32-bit* (on page 116)
- *Microsoft Windows XP 32-bit* (see "*Microsoft Windows XP 32-bit (Deprecated)*" on page 116)
- *Ubuntu 12.04 Linux 32-bit* (see "*Linux Ubuntu 12.04 LTS 32-bit*" on page 115)

5.2.2.2.2.2 In-Active Virtual Machines

5.2.2.2.2.2.1 *Fedora (Red Hat) 17 Linux 32-bit*

Fedora (Red Hat) Linux Virtual Machine (upgraded from 13 to 17)

Usability Issue(s):

- Scrolling windows plagued by image breakup
- Python 2.x and 3.x versions can be installed via Python download

5.2.2.2.2.2.2 *Mageia (Mandriva) Linux 32-bit*

Mageia (Mandriva) Linux Virtual Machine (2012/08/13)

Usability Issue(s):

- Scrolling windows plagued by image breakup

5.2.2.2.2.3 *Microsoft Windows 8 (Release Preview) 32-bit*

Microsoft Windows 8 Virtual Machine (Release Preview)

Usability Issue(s):

- Production Release expected 10/24/2012
- Supports Microsoft Office XP
- Supports Microsoft Visio XP
- Does NOT Support Author-It 5.5.0 (last version available for Windows XP through Windows 7)
- Requires Cygwin Unix-like Environment
- Python 2.x and 3.x Native Windows versions can be installed via Python download but do not support curses
- Able to establish network connection to DropBox to share latest changes to "tsWxGTUI" Toolkit

5.2.2.2.2.4 *OpenSuSE 12.2 Linux 32-bit*

OpenSuSE (12.2) Linux Virtual Machine (2012/08/09)

Usability Issue(s):

- Scrolling windows plagued by image breakup
- Unable to establish network connection to DropBox to share latest changes to "tsWxGTUI" Toolkit

5.2.2.2.2.5 *OpenSolaris (SunOS 9 & 10) UNIX 32-bit*

OpenSolaris Unix Virtual Machine (Release 9)

Usability Issue(s):

- Parallels Tools NOT available
- No built-in network connections
- Oracle/Sun no longer support this or newer versions
- Unable to establish network connection to DropBox to share latest changes to "tsWxGTUI" Toolkit

5.2.2.2.3 **VMware Fusion 5 for MAC**

VMware Fusion Desktop for Mac (upgraded from 3 through 5)

Usability Issue(s):

- Moderately configurable and user friendly (Not as polished and easy to use as Parallels).
- Can share Host resources with Guest OS (Not as versatile and easy to use as Parallels).
- Tools provided for many Guest Operating Systems. The tools are distributed as a mountable ISO-image containing an install script written in Perl source code) and B- or G-zipped Tar-files . It may be necessary to download and install various compilers and Perl tools as preparation for the VMware Fusion tool installation. The contents of the ISO image must be copied into a scratch folder on the Guest OS and extracted before the Perl manually install script is kicked off.

5.2.2.2.3.1 Active Virtual Machines

- *Cygwin (Unix-like Environment) for MS Windows* (on page 115)
- *Microsoft Windows 8.1/8 Professional 32-bit* (on page 116)
- *Microsoft Windows 7 Professional 32-bit* (on page 116)
- *Microsoft Windows XP 32-bit* (see "*Microsoft Windows XP 32-bit (Deprecated)*" on page 116)
- *Ubuntu 12.04 Linux 32-bit* (see "*Linux Ubuntu 12.04 LTS 32-bit*" on page 115)

5.2.2.2.3.1.1 Fedora 17 Linux 32-bit

Fedora (Red Hat) Linux Virtual Machine (Fedora 13 to 17)

Usability Issue(s):

- First release that is easy to download, install, configure, update and upgrade.
- Offers GNOME and KDE desktops.

5.2.2.2.3.1.2 OpenIndiana (Solaris 11 / SunOS 5.11) UNIX 32-bit

OpenIndiana (OpenSolaris) Unix Virtual Machine (SunOS 5.11)

Usability Issue(s):

- Successor to OpenSolaris
- Somewhat behind in latest Python version (2.6.4) and cannot be updated via package manager.
- Python 2.x download builds but lacks curses.
- Unable to establish network connection to DropBox to share latest changes to "tsWxGTUI" Toolkit

5.2.2.2.3.2 In-Active Virtual Machines

5.2.2.2.3.2.1 OpenSolaris (SunOS 9 & 10) UNIX 32-bit

OpenSolaris (Oracle/Sun) Unix Virtual Machine (SunOS 5.9/5.10)

Usability Issue(s):

- Oracle/Sun stopped support in 2010
- Unable to establish network connection to DropBox to share latest changes to "tsWxGTUI" Toolkit

5.2.3 Dell 15.6" Inspiron 7000 Laptop

5.2.3.1 Hardware

Dell 15.6" Inspiron 7000 Laptop:

- 366 MHz Intel Pentium II processor
- 32 KB L1 Cache
- 256 KB L2 Cache
- 384 MB RAM with 66 MHz access
- ATI RAGE Mobility Video Controller - P video with AGP 2x, 4M/8M SGRAM
- 15.6" Active-matrix color (XGA) Display (max resolution 1024 x 768 pixels; 4 billion colors)

Connectivity includes:

- One 2.5" IDE Hard Drive connector to internal 20 GB 5400 RPM 2.5" IDE Hard Drive
- Two PCMCIA Card Slots
- One USB-1.0 "low speed" port
- One PS/2 "mouse" port
- One LPT "printer" Port

Enhancements:

- 5400 RPM, 32 GB IBM 2.5" IDE Hard Disk replacement for 20 GB 5400 RPM 2.5" IDE Hard Drive
- Adaptec DuoConnect for Notebooks AUA-1422 USB-2.0/Firewire-1394 Interface to External 5400 RPM, 32 GB IBM Hard Disk
- Xircom RealPort CardBus Ethernet 10/100 Mbps + Modem 56 Kbps (used with Windows XP)
- 3Com Wireless LAN PC Card 3CRWE62092A Model WL-305 11 Mbps (used with Windows XP)
- Linksys Instant Wireless Network PC Card WPC11 version 3 100 Mbps (used with Ubuntu Linux)

Usability Issue(s):

- Does NOT support further upgrades to system hardware (RAM).
- Limited CPU speed and RAM capacity restricts upgrades to system software.

5.2.3.2 Software

5.2.3.2.1 Microsoft Windows XP 32-bit (Deprecated)

Microsoft Windows XP Professional (upgraded to SPK3)

Usability Issue(s):

- End-of-Life scheduled for 04/20/2014
- Requires Cygwin Unix-like Environment
- Supports Microsoft Office XP Professional (Deprecated)
- Supports Microsoft Visio XP Professional (Deprecated)
- Python 2.x and 3.x Native Windows versions can be installed via Python download but do not support curses
- Able to establish network connection to DropBox to share latest changes to "tsWxGTUI" Toolkit

5.2.3.2.2 Cygwin (Unix-like Environment) for MS Windows

Cygwin Unix-like Environment for MS Windows (upgraded from 1.6 to 1.8)

Usability Issue(s):

- Supports Command Line Interface with bash shell and optional xterm session.
- Optional pixel-mode Graphical-style User Interface requires X11.
- Optional character-mode Graphical-style User Interface requires nCurses terminal interface library.
- Somewhat behind in latest Python 2.x version (2.7.3) and cannot be updated via Cygwin setup nor via Python download.
- Somewhat behind in latest Python 3.x version (3.2.3) and cannot be updated via Cygwin setup nor via Python download.
- Somewhat behind in latest GNU utility versions

5.2.3.2.3 Ubuntu 12.04 Linux 32-bit

Ubuntu Linux (12.04)

Usability Issue(s):

- Marketing Claim: "Ubuntu is easy to use. And it comes with thousands of free applications. Ubuntu does everything you need it to. It'll work with your existing PC files, printers, cameras and MP3 players."
- Easy installation, configuration, enhancement and upgrade.
- Choice of UNITY, GNOME desktops, etc.
- Extensive library of optional packages available from primary (Ubuntu) and third party sources.
- Competitive and compatible with Microsoft Windows and Apple Mac OS X.
- Able to establish network connection to DropBox to share latest changes to "tsWxGTUI" Toolkit

6 DEFINITIONS, ABBREVIATIONS, AND ACRONYMS

Trademarks and copyrights (http://en.wikipedia.org/wiki/Wikipedia:About#Trademarks_and_copyrights)

Wikipedia is a registered trademark of the not-for-profit *Wikimedia Foundation*, which has created a family of free-content projects that are built by user contributions.

Most of Wikipedia's text and many of its images are dual-licensed under the *Creative Commons Attribution-Sharealike 3.0 Unported License* (CC-BY-SA) and the *GNU Free Documentation License* (GFDL) (unversioned, with no invariant sections, front-cover texts, or back-cover texts). Some text has been imported only under CC-BY-SA and CC-BY-SA-compatible license and cannot be reused under GFDL; such text is identified either on the page footer, in the page history or on the discussion page of the article that utilizes the text. Every image has a description page which indicates the license under which it is released or, if it is non-free, the rationale under which it is used.

Contributions remain the property of their creators, while the CC-BY-SA and GFDL licenses ensure the content is freely distributable and reproducible. (See the **copyright notice** (<http://en.wikipedia.org/wiki/Wikipedia:Copyrights>) and the **content disclaimer** (<http://en.wikipedia.org/wiki/Wikipedia:Disclaimers>) for more information.)

The following terms are used throughout this document:

TERM	DEFINITION
API	An application programming interface (API) is a source code based specification intended to be used as an interface by software components to communicate with each other. An API may include specifications for routines, data structures, object classes, and variables.
AuthorIT	<p>Excerpts From Wikipedia, the free encyclopedia:</p> <p>"Author-it is a help authoring tool and content management system for creating, maintaining, and distributing single-sourced content.</p> <p>Author-it can produce documentation in the following formats:</p> <ul style="list-style-type: none"> ▪ RTF, PDF, or Microsoft Word format for printed documentation ▪ Microsoft WinHelp (Windows Help) ▪ Microsoft HTML Help ▪ JavaHelp ▪ Oracle Help for Java ▪ Web and browser based help ▪ XML ▪ DITA

TERM	DEFINITION
	<p>Author-it stores all information as objects in a central database, called a library. Object types include books, topics, file objects, hyperlinks, styles, glossaries, tables of contents, indexes, and publishing profiles. The library supports multiple users. Author-it Base User module includes importing, authoring, and publishing capability. Further modules that can be licensed...."</p> <p>NOTES:</p> <ul style="list-style-type: none"> ▪ Author-it is a product of the Author-it Software Corporation, Ltd. ▪ Any chapter, section or paragraph component authored for a hierarchical location in one document (document 1 at a.b.c) may be re-used by reference at any other hierarchical location in any other document (document 2 at d.e, document 3 at f.g.h.i.j.k). Author-it automatically re-numbers new references as appropriate for their new hierarchical location. ▪ Author-it 4.5 is the last stand-alone Author-it Workgroup Edition. It uses the Microsoft JET Database Engine and is compatible with Windows XP Professional, Windows Vista and Windows 7 Professional. ▪ Author-it 5.5 is the last stand-alone Edition to use the Microsoft JET Database Engine. It is compatible with Windows XP Professional, Windows Vista, Windows 7 Professional and Windows 8 Professional. Have yet to discover how to export Microsoft JET Database to either the Microsoft SQL Server or the free Microsoft SQL Express Database. ▪ Author-it iCloud Professional and Enterprise Editions use either the Microsoft SQL Server or the free Microsoft SQL Express Database engine.
Automation	The use of various control systems for operating equipment such as machinery, processes in factories, telephone networks, steering and stabilization of ships, aircraft and other applications with minimal or reduced human intervention.
Berkley Software Distribution License	<p>From Wikipedia, the free encyclopedia</p> <p>"* * *. BSD licenses are a family of permissive free software licenses, imposing minimal restrictions on the redistribution of covered software. This is in contrast to copyleft licenses, which have reciprocity share-alike requirements. The original BSD license was used for its namesake, the Berkeley Software Distribution (BSD), a Unix-like operating system. The original version has since been revised and its descendants are more properly termed modified BSD licenses.</p> <p>Two variants of the license, the New BSD License/Modified BSD License (3-clause),[1] and the Simplified BSD License/FreeBSD License (2-clause)[2] have been verified as GPL-compatible free software licenses by the Free Software Foundation, and have been vetted as open source licenses by the Open Source Initiative,[3] while the original, 4-clause license has not been accepted as an open source license and, although the original is considered to be a free software license by the FSF, the FSF does not consider it to be compatible with the GPL due to the advertising clause.[4]"</p>
Building Blocks	A Building Block is a basic unit from which something of greater complexity is built up. For example, an application or operating system program may be constructed from one or more data structure definitions, functions, methods, classes, subroutines, threads, processes or building block libraries.
CLI	<p>Acronym for Command Line Interface.</p> <p>From Wikipedia, the free encyclopedia:</p> <p>A command-line interface (CLI) is an interface or dialog between the user and a program, or between two programs, where a line of text (a command line) is passed between the two.</p>
Communication	The application of telecommunications technology for the transmission of data to, from, or between computers over dedicated or time shared hardware.

TERM	DEFINITION
Control	<p>The application of one or more devices, to manage, command, direct or regulate the behavior of other device(s) or system(s). Industrial control systems, as used in industrial production, control equipment or machinery. In open loop control systems output is generated based only on inputs. In closed loop (feedback) control systems current output is taken into consideration and corrections are made based on feedback.</p>
Curses	<p>From Wikipedia, the free encyclopedia:</p> <p>"curses is a terminal control library for Unix-like systems, enabling the construction of text user interface (TUI) applications.name is a pun on the term "cursor optimization". It is a library of functions that manage an application's display on character-cell terminals (e.g., VT100).[1]</p> <p>Overview</p> <p>The curses API is described in several places.[2] Most implementations of curses use a database that can describe the capabilities of thousands of different terminals. There are a few implementations, such as PDCurses, which use specialized device drivers rather than a terminal database. Most implementations use terminfo; some use termcap. Curses has the advantage of back-portability to character-cell terminals and simplicity. For an application that does not require bit-mapped graphics or multiple fonts, an interface implementation using curses will usually be much simpler and faster than one using an X toolkit.</p> <p>Using curses, programmers are able to write text-based applications without writing directly for any specific terminal type. The curses library on the executing system sends the correct control characters based on the terminal type. It provides an abstraction of one or more windows that maps onto the terminal screen. Each window is represented by a character matrix. The programmer sets up each window to look as they want the display to look, and then tells the curses package to update the screen. The library determines a minimal set of changes needed to update the display and then executes these using the terminal's specific capabilities and control sequences.</p> <p>In short, this means that the programmer simply creates a character matrix of how the screen should look and lets curses handle the work."</p>

TERM	DEFINITION
Cygwin	<p>From Wikipedia, the free encyclopedia:</p> <p>"Cygwin[2] is a Unix-like environment and command-line interface for Microsoft Windows. Cygwin provides native integration of Windows-based applications, data, and other system resources with applications, software tools, and data of the Unix-like environment. Thus it is possible to launch Windows applications from the Cygwin environment, as well as to use Cygwin tools and applications within the Windows operating context.</p> <p>Cygwin consists of two parts: a dynamic-link library (DLL) as an API compatibility layer providing a substantial part of the POSIX API functionality, and an extensive collection of software tools and applications that provide a Unix-like look and feel.</p> <p>Cygwin was originally developed by Cygnus Solutions, which was later acquired by Red Hat. It is free and open source software, released under the GNU General Public License version 3. Today it is maintained by employees of Red Hat, NetApp and many other volunteers."</p> <p>See also:</p> <ul style="list-style-type: none"> ▪ Cooperative Linux ▪ Cygwin/X (X11 for Cygwin) ▪ GnuWin32 ▪ Interix ▪ MinGW (Minimalist GNU for Windows) ▪ mintty (Cygwin terminal) ▪ UWIN
Darwin	<p>From Wikipedia, the free encyclopedia:</p> <p>"Darwin is an open source Unix-like computer operating system released by Apple Inc. in 2000. It is composed of code developed by Apple, as well as code derived from NeXTSTEP, BSD, and other free software projects.</p> <p>Darwin forms the core set of components upon which OS X and iOS are based. It is mostly POSIX compatible, but has never, by itself, been certified as being compatible with any version of POSIX. (OS X, since Leopard, has been certified as compatible with the Single UNIX Specification version 3 (SUSv3).[2][3][4])</p> <p>HISTORY</p> <p>Darwin's heritage began with NeXT's NeXTSTEP operating system (later known as OpenStep), first released in 1989. After Apple bought NeXT in 1997, it announced it would base its next operating system on OpenStep. This was developed into Rhapsody in 1997, Mac OS X Server 1.0 in 1999, Mac OS X Public Beta in 2000, and Mac OS X 10.0 in 2001. In 2000, the core operating system components of Mac OS X were released as open-source software under the Apple Public Source License (APSL) as Darwin; the higher-level components, such as the Cocoa and Carbon frameworks, remained closed-source.</p> <p>Up to Darwin 8.0.1, Apple released a binary installer (as an ISO image) after each major Mac OS X release that allowed one to install Darwin on PowerPC and Intel x86 computers as a standalone operating system. Minor updates were released as packages that were installed separately. Darwin is now only available as source code,[5] except for the ARM variant, which has not been released in any form separately from iOS. However, the older versions of Darwin are still available in binary form,[6] and a hobbyist developer winocm took the official Darwin source code and ported it to ARM.[7]"</p>

TERM	DEFINITION
Debian	<p>From Wikipedia, the free encyclopedia</p> <p>"Debian (/ˈdɛbiən/) is an operating system composed primarily of free and open-source software, most of which is under the GNU General Public License, and developed by a group of individuals known as the Debian project. Debian is one of the most popular Linux distributions for personal computers and network servers, and has been used as a base for several other Linux distributions.</p> <p>Debian was first announced in 1993 by Ian Murdock, and the first stable release was made in 1996. The development is carried out over the Internet by a team of volunteers guided by a project leader and three foundational documents. New distributions are updated continually, and the next candidate is released after a time-based freeze.</p> <p>As one of the earliest Linux distributions, it was envisioned that Debian was to be developed openly in the spirit of Linux and GNU. This vision drew the attention and support of the Free Software Foundation, which sponsored the project for the first part of its life."</p>
Developer Sandbox	<p>Excerpt From Wikipedia, the free encyclopedia:</p> <p>"A sandbox is a testing environment that isolates untested code changes and outright experimentation from the production environment or repository, in the context of software development including Web development and revision control. Sandboxing protects "live" servers and their data, vetted source code distributions, and other collections of code, data and/or content, proprietary or public, from changes that could be damaging (regardless of the intent of the author of those changes) to a mission critical system or which could simply be difficult to revert. Sandboxes replicate at least the minimal functionality needed to accurately test the programs or other code under development (e.g. usage of the same environment variables as, or access to an identical database to that used by, the stable prior implementation intended to be modified; there are many other possibilities, as the specific functionality needs vary widely with the nature of the code and the application[s] for which it is intended.)</p> <p>The concept of the sandbox (sometimes also called a working directory, a test server or development server) is typically built into revision control software such as CVS and Subversion (SVN), in which developers "check out" a copy of the source code tree, or a branch thereof, to examine and work on. Only after the developer has (hopefully) fully tested the code changes in their own sandbox should the changes be checked back into and merged with the repository and thereby made available to other developers or end users of the software.[1]</p> <p>By further analogy, the term "sandbox" can also be applied in computing and networking to other temporary or indefinite isolation areas, such as security sandboxes and search engine sandboxes (both of which have highly specific meanings), that prevent incoming data from affecting a "live" system (or aspects thereof) unless/until defined requirements or criteria have been met."</p>
Diagnostic	<p>The application of technology to locate problems with software, hardware, or any combination thereof in a system, or a network of systems. Diagnostics typically provide guidance to the user to solve issues.</p>
Docstring	<p>Excerpt from http://www.python.org/dev/peps/pep-0257/:</p> <p>"A docstring is a string literal that occurs as the first statement in a module, function, class, or method definition. Such a docstring becomes the <code>__doc__</code> special attribute of that object.</p> <p>All modules should normally have docstrings, and all functions and classes exported by a module should also have docstrings. Public methods (including the <code>__init__</code> constructor) should also have docstrings. A package may be documented in the module docstring of the <code>__init__.py</code> file in the package directory.</p> <p>String literals occurring elsewhere in Python code may also act as documentation. They are not recognized by the Python bytecode compiler and are not accessible as runtime object attributes (i.e.</p>

TERM	DEFINITION
	<p>not assigned to <code>__doc__</code>), but two types of extra docstrings may be extracted by software tools:</p> <ul style="list-style-type: none"> ▪ String literals occurring immediately after a simple assignment at the top level of a module, class, or <code>__init__</code> method are called "attribute docstrings". ▪ String literals occurring immediately after another docstring are called "additional docstrings". <p>Please see PEP 258, "Docutils Design Specification" [2], for a detailed description of attribute and additional docstrings."</p>
Dropbox	<p>From www.dropbox.com:</p> <p>"Dropbox is a free service that lets you bring all your photos, docs, and videos anywhere. Any file you save to your Dropbox will also automatically save to all your computers, phones, and even the Dropbox website. This means that you can start working on your computer at school or the office, and finish on your home computer. Never email yourself a file again!"</p>
Extension Module	<p>A module written in the low-level language of the Python implementation: C/C++ for Python, Java for Jython. Typically contained in a single dynamically loadable pre-compiled file, e.g. a shared object (.so) file for Python extensions on Unix, a DLL (given the .pyd extension) for Python extensions on Windows, or a Java class file for Jython extensions. (Note that currently, the Distutils only handles C/C++ extensions for Python.)</p>
FreeBSD	<p>From Wikipedia, the free encyclopedia:</p> <p>"FreeBSD is a free Unix-like operating system descended from Research Unix via Berkeley Software Distribution (BSD). Although for legal reasons FreeBSD cannot use the Unix trademark, it is a direct descendant of BSD, which was historically also called "BSD Unix" or "Berkeley Unix". The first version of FreeBSD was released in 1993, and today FreeBSD is the most widely used open-source BSD distribution, accounting for more than three-quarters of all installed systems running open-source BSD derivatives.[3]</p> <p>FreeBSD has similarities with Linux, with two major differences in scope and licensing: FreeBSD maintains a complete operating system, i.e. the project delivers kernel, device drivers, userland utilities and documentation, as opposed to a kernel only;[4] and FreeBSD source code is generally released under a permissive BSD license as opposed to the more restrictive GPL.</p> <p>The FreeBSD project includes a security team overlooking all software shipped in the base distribution. A wide range of additional third-party applications may be installed via two package managers, "pkgng" and the FreeBSD Ports, or by directly compiling source code. Due to its permissive licensing terms, much of FreeBSD's code base has become an integral part of other operating systems such as Juniper JUNOS and Apple's OS X."</p>

TERM	DEFINITION
Functional Requirements	<p>From Wikipedia, the free encyclopedia:</p> <p>"In software engineering (and System Engineering), a functional requirement defines a function of a system or its component. A function is described as a set of inputs, the behavior, and outputs (see also software). Functional requirements may be calculations, technical details, data manipulation and processing and other specific functionality that define what a system is supposed to accomplish. Behavioral requirements describing all the cases where the system uses the functional requirements are captured in use cases. Functional requirements are supported by non-functional requirements (also known as quality requirements), which impose constraints on the design or implementation (such as performance requirements, security, or reliability). Generally, functional requirements are expressed in the form "system must do <requirement>", while non-functional requirements are "system shall be <requirement>". The plan for implementing functional requirements is detailed in the system design. The plan for implementing non-functional requirements is detailed in the system architecture.</p> <p>As defined in requirements engineering, functional requirements specify particular results of a system. This should be contrasted with non-functional requirements which specify overall characteristics such as cost and reliability. Functional requirements drive the application architecture of a system, while non-functional requirements drive the technical architecture of a system.</p> <p>In some cases a requirements analyst generates use cases after gathering and validating a set of functional requirements. The hierarchy of functional requirements is: user/stakeholder request → feature → use case → business rule. Each use case illustrates behavioral scenarios through one or more functional requirements. Often, though, an analyst will begin by eliciting a set of use cases, from which the analyst can derive the functional requirements that must be implemented to allow a user to perform each use case."</p>
GNU	<p>From Wikipedia, the free encyclopedia:</p> <p>"GNU [2][3] is a Unix-like computer operating system developed by the GNU Project, ultimately aiming to be a "complete Unix-compatible software system" [1][4][5] composed wholly of free software. Development of GNU was initiated by Richard Stallman in 1983 [1][6] and was the original focus of the Free Software Foundation (FSF). [1][7][8][9] Non-GNU kernels, most famously the Linux kernel, can also be used with GNU. [10][11][12] The FSF maintains that Linux, when used with GNU tools and utilities, should be considered a variant of GNU, and promotes the term Linux for such systems (leading to the Linux naming controversy). [13][14][15]</p> <p>GNU is a recursive acronym for "GNU's Not Unix!", [1][16] chosen because GNU's design is Unix-like, but differs from Unix by being free software and containing no Unix code. [17] Programs released under the auspices of the GNU Project are called GNU packages or GNU programs. The system's basic components include the GNU Compiler Collection (GCC), the GNU C library (glibc), and GNU Core Utilities (coreutils), [1] but also the GNU Debugger (GDB), GNU Binary Utilities (binutils), and the bash shell. [18] GNU developers have contributed GNU/Linux Linux ports of GNU applications and utilities, which are now also widely used on other operating systems such as BSD variants, Solaris and Mac OS X. [19]</p> <p>The GNU General Public License (GPL), the GNU Lesser General Public License (LGPL), and the GNU Free Documentation License (GFDL) were written for GNU and the GNU Affero General Public License was written as an extended version of GPL version 3 for programs run over a network, but the GNU Project's licenses are also used by many unrelated projects. A minority of the software used by GNU, such as the X Window System, is licensed under permissive free software licenses.</p> <p>Richard Stallman views GNU as a "technical means to a social end".[20]"</p>
GNU/Linux	Excerpted From https://www.gnu.org/gnu/linux-and-gnu.html :

TERM	DEFINITION
	<p>"Linux and the GNU System by Richard Stallman</p> <p>For more information see also the GNU/Linux FAQ, and Why GNU/Linux?</p> <p>Many computer users run a modified version of the GNU system every day, without realizing it. Through a peculiar turn of events, the version of GNU which is widely used today is often called "Linux", and many of its users are not aware that it is basically the GNU system, developed by the GNU Project.</p> <p>There really is a Linux, and these people are using it, but it is just a part of the system they use. Linux is the kernel: the program in the system that allocates the machine's resources to the other programs that you run. The kernel is an essential part of an operating system, but useless by itself; it can only function in the context of a complete operating system. Linux is normally used in combination with the GNU operating system: the whole system is basically GNU with Linux added, or GNU/Linux. All the so-called "Linux" distributions are really distributions of GNU/Linux.</p> <p>Many users do not understand the difference between the kernel, which is Linux, and the whole system, which they also call "Linux". The ambiguous use of the name doesn't help people understand. These users often think that Linus Torvalds developed the whole operating system in 1991, with a bit of help.</p> <p>Programmers generally know that Linux is a kernel. But since they have generally heard the whole system called "Linux" as well, they often envisage a history that would justify naming the whole system after the kernel. For example, many believe that once Linus Torvalds finished writing Linux, the kernel, its users looked around for other free software to go with it, and found that (for no particular reason) most everything necessary to make a Unix-like system was already available.</p> <p>What they found was no accident: it was the not-quite-complete GNU system. The available free software added up to a complete system because the GNU Project had been working since 1984 to make one. In the The GNU Manifesto we set forth the goal of developing a free Unix-like system, called GNU. The Initial Announcement of the GNU Project also outlines some of the original plans for the GNU system. By the time Linux was started, GNU was almost finished.</p> <p>Most free software projects have the goal of developing a particular program for a particular job. For example, Linus Torvalds set out to write a Unix-like kernel (Linux); Donald Knuth set out to write a text formatter (TeX); Bob Scheifler set out to develop a window system (the X Window System). It's natural to measure the contribution of this kind of project by specific programs that came from the project.</p> <p>If we tried to measure the GNU Project's contribution in this way, what would we conclude? One CD-ROM vendor found that in their "Linux distribution", GNU software was the largest single contingent, around 28% of the total source code, and this included some of the essential major components without which there could be no system. Linux itself was about 3%. (The proportions in 2008 are similar: in the "main" repository of gNewSense, Linux is 1.5% and GNU packages are 15%.) So if you were going to pick a name for the system based on who wrote the programs in the system, the most appropriate single choice would be "GNU".</p> <p>But that is not the deepest way to consider the question. The GNU Project was not, is not, a project to develop specific software packages. It was not a project to develop a C compiler, although we did that. It was not a project to develop a text editor, although we developed one. The GNU Project set out to develop a complete free Unix-like system: GNU."</p> <p><i>The complete article is available at the aforementioned link.</i></p>
gnuwin32	<p>From Wikipedia, the free encyclopedia:</p> <p>"The GnuWin32 project provides native ports in the form of runnable computer programs, patches, and source code for various GNU and open source tools and software, much of it modified to run on</p>

TERM	DEFINITION
	<p>the 32-bit Windows platform. The ports included in the GnuWin32 packages are:</p> <ul style="list-style-type: none"> ▪ GNU utilities such as bc, bison, chess, Coreutils, diffutils, ed, Flex, gawk, gettext, grep, Groff, gzip, iconv, less, m4, patch, readline, rx, sharutils, sed, tar, texinfo, units, Wget, which ▪ Archive management and compression tools, such as: arc, arj, bzip2, gzip, lha, zip, zlib. ▪ Non-GNU utilities such as: cygutils, file, ntfsprogs, OpenSSL, PCRE. ▪ Graphics tools. ▪ PDCurses ▪ Tools for processing text. ▪ Mathematical software and statistics Software." <p>See also:</p> <ul style="list-style-type: none"> ▪ Cygwin ▪ DJGPP ▪ GNUWin II ▪ Microsoft Windows Services for UNIX ▪ MinGW, MSYS ▪ UnxUtils ▪ UWIN
GUI	<p>Acronym for Graphical User Interface.</p> <p>From Wikipedia, the free encyclopedia:</p> <p>"In computing, a graphical user interface (GUI, commonly pronounced gooey[1]) is a type of user interface that allows users to interact with electronic devices with images rather than text commands. GUIs can be used in computers, hand-held devices such as MP3 players, portable media players or gaming devices, household appliances and office equipment. A GUI represents the information and actions available to a user through graphical icons and visual indicators such as secondary notation, as opposed to text-based interfaces, typed command labels or text navigation. The actions are usually performed through direct manipulation of the graphical elements.[2]"</p>
High Level API	<p>A programming interface provides more functionality within one statement than a lower-level interface. High-level interfaces are designed to enable the programmer to write code in a shorter amount of time and to be less involved with the details of the software module or hardware device that is performing the required services.</p>
Hypervisor	<p>From Wikipedia, the free encyclopedia</p> <p>"In computing, a hypervisor, also called virtual machine manager (VMM), is one of many hardware virtualization techniques allowing multiple operating systems, termed guests, to run concurrently on a host computer. It is so named because it is conceptually one level higher than a supervisory program. The hypervisor presents to the guest operating systems a virtual operating platform and manages the execution of the guest operating systems. Multiple instances of a variety of operating systems may share the virtualized hardware resources. Hypervisors are very commonly installed on server hardware, with the function of running guest operating systems, that themselves act as servers.</p> <p>The term can be used to describe the interface provided by the specific cloud computing functionality infrastructure as a service (IaaS).[1][2]</p> <p>The term "hypervisor" was first used in 1965, referring to software that accompanied an IBM RPQ for the IBM 360/65. It allowed the model IBM 360/65 to share its memory: half acting as an IBM</p>

TERM	DEFINITION
	360 and half as an emulated IBM 7080. The software, labeled "hypervisor," did the switching between the two modes on split-time basis. The term hypervisor was coined as an evolution of the term "supervisor," the software that provided control on earlier hardware.[3][4]"
Instrumentation	The application of technology for the measurement and control of process variables within a production or manufacturing area. An instrument is a device that measures a physical quantity such as flow, temperature, level, distance, angle, or pressure. Instruments may be as simple as direct reading thermometers or may be complex multi-variable process analyzers. Instruments are often part of a control system in refineries, factories, and vehicles.
Interface Requirements Specification	The Interface Requirements Specification (IRS) specifies the requirements imposed on one or more systems, subsystems, Hardware Configuration Items (HWCIs), Computer Software Configuration Items (CSCIs), manual operations, or other system components to achieve one or more interfaces among these entities. An IRS can cover any number of interfaces.
Linux	<p>From Wikipedia, the free encyclopedia</p> <p>"This article is about the operating system. For the kernel, see Linux kernel.</p> <p>Linux [8][9][10] is a Unix-like computer operating system assembled under the model of free and open source software development and distribution. The defining component of GNU/Linux Linux is the Linux kernel, an operating system kernel first released 5 October 1991 by Linus Torvalds.[11][12]"</p> <p>NOTE:</p> <ul style="list-style-type: none"> ▪ The Free Software Foundation (FSF) is responsible for the GNU Project and maintains that Linux Linux, when used with GNU tools and utilities, should be considered a variant of GNU, and promotes the term Linux for such systems (leading to the Linux naming controversy).
Low Level API	A programming interface that is the most detailed, allowing the programmer to manipulate functions within a software module or within hardware at a very granular level.
MAC OS X	<p>From Wikipedia, the free encyclopedia</p> <p>"Mac OS is a series of graphical user interface-based operating systems developed by Apple Inc. (formerly Apple Computer, Inc.) for their Macintosh line of computer systems. Mac OS is credited with popularizing the graphical user interface. The original form of what Apple would later name the "Mac OS" (currently OSX) was the integral and unnamed system software first introduced in 1984 with the original Macintosh, usually referred to simply as the System software."</p>
ManPage	A manpage (short for manual page) is a form of online software documentation usually found on a Unix or Unix-like operating system. Topics covered include computer programs (including library and system calls), formal standards and conventions, and even abstract concepts.
Massachusetts Institute of Technology License	<p>From Wikipedia, the free encyclopedia</p> <p>"The MIT License is a free software license originating at the Massachusetts Institute of Technology (MIT). It is a permissive free software license, meaning that it permits reuse within proprietary software provided all copies of the licensed software include a copy of the MIT License terms. Such proprietary software retains its proprietary nature even though it incorporates software under the MIT License. The license is also GPL-compatible, meaning that the GPL permits combination and redistribution with software that uses the MIT License.[1]</p> <p>Notable software packages that use one of the versions of the MIT License include Expat, PuTTY, the Mono development platform class libraries, Ruby on Rails, Lua (from version 5.0 onwards), Wayland and the X Window System, for which the license was written."</p>

TERM	DEFINITION
Microsoft Windows	<p>From Wikipedia, the free encyclopedia</p> <p>"Microsoft Windows is a series of graphical interface operating systems developed, marketed, and sold by Microsoft.</p> <p>Microsoft introduced an operating environment named Windows on November 20, 1985 as an add-on to MS-DOS in response to the growing interest in graphical user interfaces (GUIs).[2] Microsoft Windows came to dominate the world's personal computer market with over 90% market share, overtaking Mac OS, which had been introduced in 1984."</p>
MIL-STD-498	<p>From Wikipedia, the free encyclopedia</p> <p>"MIL-STD-498 (Military-Standard-498) was a United States military standard whose purpose was to "establish uniform requirements for software development and documentation." It was released Nov. 8, 1994, and replaced DOD-STD-2167A, DOD-STD-7935A, and DOD-STD-1703. It was meant as an interim standard, to be in effect for about two years until a commercial standard was developed.</p> <p>Unlike previous efforts like the seminal "2167A" which was mainly focused on the risky new area of software development, "498" was the first attempt at a truly comprehensive description of the systems development life-cycle. It was the baseline that all of the ISO, IEEE, and related efforts after it replaced. It also contains much of the material that the subsequent professionalization of project management covered in the Project Management Body of Knowledge (PMBOK). The document "MIL-STD-498 Overview and Tailoring Guidebook" is 98 pages. The "MIL-STD-498 Application and Reference Guidebook" is 516 pages. Associated to these were document templates, or Data Item Descriptions, described below, bringing documentation and process order that could scale to projects of the size humans were then conducting (aircraft, battleships, canals, dams, factories, satellites, submarines, etcetera).</p> <p>It was one of the few military standards that survived the "Perry Memo", then U.S. Secretary of Defense William Perry's 1994 memorandum commanding the discontinuation of defense standards. However, it was canceled on May 27, 1998 and replaced by the essentially identical demilitarized version EIA J-STD-016[1] [2] as a process example guide for IEEE 12207. Several programs outside of the U.S. military continued to use the standard due to familiarity and perceived advantages over alternative standards, such as free availability of the standards documents and presence of process detail including contractually-usable Data Item Descriptions."</p> <hr/> <p>From http://everyspec.com/MIL-STD/MIL-STD-0300-0499/MIL-STD-498_25500/</p> <p>"MIL-STD-498, MILITARY STANDARD: SOFTWARE DEVELOPMENT AND DOCUMENTATION (05 DEC 1994) [SUPERSEDING MIL-STD-2167A, DOD-STD-7935A & DOD-STD-1703] [S/S BY IEEE/EIA 12207.0, IEEE/EIA 12207.1 & IEEE/EIA 12207.2]., The purpose of this standard is to establish uniform requirements for software development and documentation. This standard merges DOD-STD-2167A and DOD-STD-7935A to define a set of activities and documentation suitable for the development of both weapon systems and Automated Information Systems. A conversion guide from these standards to MIL-STD-498 is provided in Appendix I. Other changes include improved compatibility with incremental and evolutionary development models; improved compatibility with non-hierarchical design methods; improved compatibility with computer-aided software engineering (CASE) tools; alternatives to, and more flexibility in, preparing documents; clearer requirements for incorporating reusable software; introduction of software management indicators; added emphasis on software supportability; and improved links to systems engineering. This standard supersedes DOD-STD-2167A, DOD-STD- 7935A, and DOD-STD-1703</p>

TERM	DEFINITION
	(NS)." <u>A copy of the specification is available via a download link.</u>
Module	The basic unit of code reusability in Python: a block of code imported by some other code. Three types of modules concern us here: pure Python modules, extension modules, and packages.
nCurses	From Wikipedia, the free encyclopedia "ncurses (new curses) is a programming library that provides an API which allows the programmer to write text-based user interfaces in a terminal-independent manner. It is a toolkit for developing "GUI-like" application software that runs under a terminal emulator. It also optimizes screen changes, in order to reduce the latency experienced when using remote shells."
Non-Functional Requirements	From Wikipedia, the free encyclopedia: See Functional Requirements for definition and relationship.
Notebooks	A collection of commentaries that express opinions or offerings of explanations about events or situations that might be useful to Toolkit installers, developers, operators, troubleshooters and distributors. The documents may be in Application-specific formats (Adobe PDF, JPEG Bit-mapped image, Microsoft Office, Plain text etc.).
OpenIndiana	From Wikipedia, the free encyclopedia "OpenIndiana is a Unix-like computer operating system released as free and open source software. It forked from OpenSolaris after the discontinuation of that project by Oracle[1] and aims to continue development and distribution of the OpenSolaris codebase.[2] The project operates under the umbrella of the Illumos Foundation.[2] The stated aim of the project is "[...] to become the defacto OpenSolaris distribution installed on production servers where security and bug fixes are required free of charge".[3]"
OpenSolaris	From Wikipedia, the free encyclopedia "OpenSolaris (...) was an open source computer operating system based on Solaris created by Sun Microsystems. It was also the name of the project initiated by Sun to build a developer and user community around the software. After the acquisition of Sun Microsystems in 2010, Oracle decided to discontinue open development of the core software, and replaced the OpenSolaris distribution model with the proprietary Solaris Express. Prior to Oracle's moving of core development "behind closed doors", a group of former OpenSolaris developers decided to "fork" the core software under the name OpenIndiana. The project, a part of the Illumos Foundation, aims to continue the development and distribution of the OpenSolaris codebase.[5] OpenSolaris is a descendant of the UNIX System V Release 4 (SVR4) code base developed by Sun and AT&T in the late 1980s. It is the only version of the System V variant of UNIX available as open source.[6] OpenSolaris is developed as a combination of several software consolidations that were open sourced subsequent to Solaris 10. It includes a variety of free software, including popular desktop and server software.[7][8] On Friday, August 13, 2010, details started to emerge relating to the restructuring of the OpenSolaris project, the pending release of the new future commercial version of Solaris, Solaris 11, and how open source community interactions are being adjusted.[9]"
Package	A module that contains other modules; typically contained in a directory in the filesystem and distinguished from other directories by the presence of a file <code>__init__.py</code> .
Parallels Desktop for Mac	From Wikipedia, the free encyclopedia "Parallels Desktop for Mac by Parallels, Inc., is software providing hardware virtualization for

TERM	DEFINITION
	<p>Macintosh computers with Intel processors.</p> <p>Technical</p> <p>Parallels Desktop for Mac is a hardware emulation virtualization software, using hypervisor technology that works by mapping the host computer's hardware resources directly to the virtual machine's resources. Each virtual machine thus operates identically to a standalone computer, with virtually all the resources of a physical computer.[3] Because all guest virtual machines use the same hardware drivers irrespective of the actual hardware on the host computer, virtual machine instances are highly portable between computers. For example, a running virtual machine can be stopped, copied to another physical computer, and restarted."</p> <p>Parallels Tools</p> <p>Parallels Tools are a suite of behind-the-scenes tools that allow seamless operating between Mac OS X and Windows or another guest operating system.</p> <p>Each Parallels release includes its own Parallels Tools. Those tools interface the Guest Operating System's Virtual Machine to the Parallels Desktop installed on the host computer. The Tools enable the Host and Guest OS to share resources.</p> <p>After upgrading the Parallels Desktop from 9 to 10 on one host computer it was still possible to run recent copies of the Paralels 10 Guest OS virtual machines on a host computer that could not be upgraded to Parallels 10 simply by:</p> <ul style="list-style-type: none"> ▪ Attaching a hard drive with the Parallels 10 Guest OS virtual machine. ▪ Manually changing the Parallels Guest OS configuration to suit the available host memory and devices. ▪ Allowing Parallels to automatically (or manually initiating) re-install of the available older Parallels (8 or 9) Tools. ▪ Launching the re-configured Parallels Guest OS.
PC-BSD or PCBSD	<p>From Wikipedia, the free encyclopedia</p> <p>"PC-BSD, or PCBSD, is a Unix-like, desktop-oriented operating system built upon the most recent releases of FreeBSD. It aims to be easy to install by using a graphical installation program, and easy and ready-to-use immediately by providing KDE SC, LXDE, Xfce, and MATE [1] as the graphical user interface. It provides official binary nVidia and Intel drivers for hardware acceleration and an optional 3D desktop interface through Kwin, and Wine is ready-to-use in running Microsoft Windows software. PC-BSD is able to run Linux software,[2] in addition to FreeBSD ports, and it has its own package management system that allows users to graphically install pre-built software packages from a single downloaded executable file, which is unique for BSD operating systems.</p> <p>PC-BSD supports ZFS, and the installer offers disk encryption with geli so the system will require a passphrase before booting."</p>
PDCurses	<p>PDCurses is an implementation of the curses library for X11. It provides the ability for existing text-mode curses programs to be re-built as native X11 applications with very little modification. PDCurses for X11 is also known as XCurses.</p> <p>It is available from http://pdcurses.sourceforge.net. Version 2.6 enables Python applications launched within the Windows Command Prompt shell to import and use the Python Curses module.</p> <p>However, Version 2.6 cannot be used by the "tsWxGraphicalTextUserInterface" module to emulate "wxPython" because it lacks the mouse button definitions and interface features available with Unix-like shells such as bash.</p>
POSIX	<p>From Wikipedia, the free encyclopedia</p>

TERM	DEFINITION
	<p>"Not to be confused with Unix, Unix-like, or Linux.</p> <p>An acronym for "Portable Operating System Interface", is a family of standards specified by the IEEE for maintaining compatibility between operating systems. POSIX defines the application programming interface (API), along with command line shells and utility interfaces, for software compatibility with variants of Unix and other operating systems.[1][2]"</p>
Programming Tools or Software Development Tools	<p>From Wikipedia, the free encyclopedia</p> <p>"A programming tool or software development tool is a computer program that software developers use to create, debug, maintain, or otherwise support other programs and applications. The term usually refers to relatively simple programs, that can be combined together to accomplish a task, much as one might use multiple hand tools to fix a physical object. The ability to use a variety of tools productively is one hallmark of a skilled software engineer.</p> <p>The most basic tools are a source code editor and a compiler or interpreter, which are used ubiquitously and continuously. Other tools are used more or less depending on the language, development methodology, and individual engineer, and are often used for a discrete task, like a debugger or profiler. Tools may be discrete programs, executed separately – often from the command line – or may be parts of a single large program, called an integrated development environment (IDE). In many cases, particularly for simpler use, simple ad hoc techniques are used instead of a tool, such as print debugging instead of using a debugger, manual timing (of overall program or section of code) instead of a profiler, or tracking bugs in a text file or spreadsheet instead of a bug tracking system.</p> <p>The distinction between tools and applications is murky. For example, developers use simple databases (such as a file containing a list of important values) all the time as tools.[dubious – discuss] However a full-blown database is usually thought of as an application or software in its own right. For many years, computer-assisted software engineering (CASE) tools were sought after. Successful tools have proven elusive.[citation needed] In one sense, CASE tools emphasized design and architecture support, such as for UML. But the most successful of these tools are IDEs."</p>
Pure Python Module	<p>A module written in Python and contained in a single .py file (and possibly associated .pyc and/or .pyo files). Sometimes referred to as a "pure module."</p>
Python	<p>From Wikipedia, the free encyclopedia:</p> <p>"Python is a general-purpose, high-level programming language[11] whose design philosophy emphasizes code readability.[12] Python claims to combine "remarkable power with very clear syntax",[13] and its standard library is large and comprehensive.</p> <p>Python supports multiple programming paradigms, primarily but not limited to object-oriented, imperative and, to a lesser extent, functional programming styles. It features a fully dynamic type system and automatic memory management, similar to that of Scheme, Ruby, Perl, and Tcl. Like other dynamic languages, Python is often used as a scripting language, but is also used in a wide range of non-scripting contexts. Using third-party tools, Python code can be packaged into standalone executable programs. Python interpreters are available for many operating systems.</p> <p>The reference implementation of Python (CPython) is free and open source software and has a community-based development model, as do all or nearly all of its alternative implementations. CPython is managed by the non-profit Python Software Foundation."</p>
Python Software Foundation License	<p>From Wikipedia, the free encyclopedia:</p> <ul style="list-style-type: none"> ▪ "Python Software Foundation License FSF approved Yes [1] ▪ OSI approved Yes ▪ GPL compatible Yes [1]

TERM	DEFINITION
	<p>▪ Copyleft No</p> <p>The Python Software Foundation License (PSFL) is a BSD-style, permissive free software license which is compatible with the GNU General Public License (GPL).[1] Its primary use is for distribution of the Python project software. Unlike the GPL the Python license is not a copyleft license, and allows modifications to the source code, as well as the construction of derivative works, without making the code open-source. The PSFL is listed as approved on both FSF's approved licenses list,[1] and OSI's approved licenses list.</p> <p>Earlier versions of Python were under the so-called Python License, which is incompatible with the GPL. The reason given for this incompatibility by Free Software Foundation was that "this Python license is governed by the laws of the 'State of Virginia', in the USA", and the GPL does not permit this.[2]</p> <p>The year that Python's creator Guido van Rossum changed the license to fix this incompatibility, he was awarded the Free Software Foundation Award for the Advancement of Free Software.[3]"</p>
Python Virtual Machine	<p>A virtual machine designed to interpret and execute programs implemented in the Python programming language.</p> <p>From Wikipedia, the free encyclopedia:</p> <p>"A virtual machine (VM) is a software implementation of a machine (i.e. a computer) that executes programs like a physical machine. Virtual machines are separated into two major categories, based on their use and degree of correspondence to any real machine. A system virtual machine provides a complete system platform which supports the execution of a complete operating system (OS). In contrast, a process virtual machine is designed to run a single program, which means that it supports a single process. An essential characteristic of a virtual machine is that the software running inside is limited to the resources and abstractions provided by the virtual machine—it cannot break out of its virtual world.</p> <p>A virtual machine was originally defined by Popek and Goldberg as "an efficient, isolated duplicate of a real machine". Current use includes virtual machines which have no direct correspondence to any real hardware.[2]"</p>
Repository	<p>Excerpt From Wikipedia, the free encyclopedia:</p> <p>"A software repository is a storage location from which software packages may be retrieved and installed on a computer."</p> <p>The TeamSTARS "tsWxGTUI_PyVx" Toolkit repository is the collection of documentation and computer program source code files that is being distributed by its author in order to publish and share the intellectual property with others.</p>
Root Package	<p>The root of the hierarchy of packages. (This isn't really a package, since it doesn't have an <code>__init__.py</code> file. But we have to call it something.) The vast majority of the standard library is in the root package, as are many small, standalone third-party modules that don't belong to a larger module collection. Unlike regular packages, modules in the root package can be found in many directories: in fact, every directory listed in <code>sys.path</code> contributes modules to the root package.</p>
Simulation	<p>The application of technology for the imitation of the operation of a real-world process or system over time. The act of simulating something first requires that a model be developed; this model represents the key characteristics or behaviors/functions of the selected physical or abstract system or process. The model represents the system itself, whereas the simulation represents the operation of the system over time.</p>

TERM	DEFINITION
Site-Package	<p>The location where third-party packages are installed (i.e., those not part of the core Python distribution). NOTE: That with Linux, Mac OS X and Unix operating systems one must have root privileges to write to that location.</p> <p>Unlike the contents of the Developer-Sandbox, the third-party site-package and it users must explicitly import via the site-package.package.module path identifier.</p>
Software Engineer	<p>An individual who will specify, plan, supervise and/or perform the design, development, coding, testing, debugging, maintenance and support of application programs, with Command Line Interface or Graphical User Interface, to be used by System Operators.</p> <p>The term also applies to those individuals who perform similar engineering activities associated with communication, data base, simulation, operating system, device driver, test and diagnostic components.</p>
Software Requirements Specification	<p>The Software Requirements Specification (SRS) specifies the requirements for a Computer Software Configuration Item (CSCI) and the methods to be used to ensure that each requirement has been met. Requirements pertaining to the CSCI's external interfaces may be presented in the SRS or in one or more Interface Requirements Specifications (IRSs) (DI-IPSC-81434) referenced from the SRS.</p>
Solaris	<p>From Wikipedia, the free encyclopedia</p> <p>"Solaris is a Unix operating system originally developed by Sun Microsystems. It superseded their earlier SunOS in 1993. Oracle Solaris, as it is now known, has been owned by Oracle Corporation since Oracle's acquisition of Sun in January 2010.[2]</p> <p>Solaris is known for its scalability, especially on SPARC systems, and for originating many innovative features such as DTrace, ZFS and Time Slider.[3][4] Solaris supports SPARC-based and x86-based workstations and servers from Sun and other vendors, with efforts underway to port to additional platforms. Solaris is registered as compliant with the Single Unix Specification.</p> <p>Solaris was historically developed as proprietary software, then in June 2005 Sun Microsystems released most of the codebase under the CDDL license, and founded the OpenSolaris open source project.[5] With OpenSolaris, Sun wanted to build a developer and user community around the software. After the acquisition of Sun Microsystems in January 2010, Oracle decided to discontinue the OpenSolaris distribution and the development model.[6][7] Just ten days before the internal Oracle memo announcing this decision to employees was "leaked", Garrett D'Amore had announced[8] the illumos project, creating a fork of the Solaris kernel and launching what has since become a thriving alternative to Oracle Solaris.</p> <p>In August 2010, Oracle discontinued providing public updates to the source code of the Solaris Kernel, effectively turning Solaris 11 into a closed source proprietary operating system. However, through the Oracle Technology Network (OTN), industry partners can still gain access to the in-development Solaris source code.[7] The Open source portion of Solaris 11 is available for download from Oracle.[9]"</p>

TERM	DEFINITION
Source Code	<p>Excerpt From Wikipedia, the free encyclopedia:</p> <p>"In computing, source code is any collection of computer instructions (possibly with comments) written using some human-readable computer language, usually as text. The source code of a program is specially designed to facilitate the work of computer programmers, who specify the actions to be performed by a computer mostly by writing source code. The source code is often transformed by a compiler program into low-level machine code understood by the computer. The machine code might then be stored for execution at a later time. Alternatively, an interpreter can be used to analyze and perform the outcomes of the source code program directly on the fly.</p> <p>Most computer applications are distributed in a form that includes executable files, but not their source code. If the source code were included, it would be useful to a user, programmer, or system administrator, who may wish to modify the program or to understand how it works.</p> <p>Aside from its machine-readable forms, source code also appears in books and other media; often in the form of small code snippets, but occasionally complete code bases; a well-known case is the source code of PGP."</p>
Stakeholder	Those persons, groups or organizations with an interest in a project.
System Administration Utilities	Computer programs that computer system administrators use to install, debug, maintain, or otherwise support computer hardware and software.
System Administrator	An individual who will specify, plan, supervise and/or perform the installation, configuration, maintenance, repair and support of the computer hardware, operating system, application software and communication network to be used by Software Engineers and System Operators.
System Operator	An individual who will use various application programs, with associated Command Line Interface or Graphical User Interface, to interactively supervise communication, control, data base, diagnostic, instrumentation or simulation activities.
TDD	<p>Acronym for Test-Driven Development</p> <p>from: http://encyclopedia.thefreedictionary.com/Test+Driven+Development</p> <p>"Test-driven development (TDD) is a software development process that relies on the repetition of a very short development cycle: first the developer writes a failing automated test case that defines a desired improvement or new function, then produces code to pass that test and finally refactors the new code to acceptable standards. Kent Beck, who is credited with having developed or 'rediscovered' the technique, stated in 2003 that TDD encourages simple designs and inspires confidence.[1]</p> <p>Test-driven development is related to the test-first programming concepts of extreme programming, begun in 1999,[2] but more recently has created more general interest in its own right.[3]</p> <p>Programmers also apply the concept to improving and debugging legacy code developed with older techniques.[4]"</p>
tsToolKitCLI	<p>The identifier for a package of toolkit components for a Python-based Command Line Interface. The library is further subdivided into the following components:</p> <ul style="list-style-type: none"> ▪ tsLibCLI - library of command line building blocks that establishes the Unix-style, run time environment for pre-processing source files, launching application programs, handling events (registering events with date, time and event severity annotations) and configuring console terminal and file system input and output. ▪ tsTestsCLI - a set of command line interface application programs and scripts for regression testing and tutorial demos.

TERM	DEFINITION
	<ul style="list-style-type: none"> ▪ tsToolsCLI - a set of command line interface application programs and utility scripts for: checking source code syntax and style via "plint"; generating Unix-style "man page" documentation from source code comments via "pydoc"; and installing, modifying for publication, and tracking software development metrics. ▪ tsUtilities - a library of computer system configuration, diagnostic, installation, maintenance and performance tool components for various host hardware and software platforms.
Terminal Emulator	<p>Excerpt from Wikipedia, the free encyclopedia</p> <p>"A program that emulates a video terminal within some other display architecture. Though typically synonymous with a shell or text terminal, the term terminal covers all remote terminals, including graphical interfaces. A terminal emulator inside a graphical user interface is often called a terminal window."</p>
tsToolkitGUI	<p>The identifier for a package of toolkit components for a "wxPython"-style, "nCurses"-based Graphical-style User Interface. The library is further subdivided into the following components:</p> <ul style="list-style-type: none"> ▪ tsLibGUI - a library of graphical-style user interface building blocks that establishes the emulated run time environment for the high-level, pixel-mode, "wxPython" GUI Toolkit via the low-level, character-mode, "nCurses" Text User Interface Toolkit ▪ tsTestsGUI - a set of graphical-style user interface application programs for regression testing and tutorial demos. ▪ tsToolsGUI - a set of graphical-style user interface application programs for tracking software development metrics.
tsLibCLI	The identifier for a library of building-block components for a Python-based Command Line Interface.
tsLibGUI	The identifier for a library of building-block components for a "wxPython"-style, "nCurses"-based Graphical-style User Interface.
tsTestsCLI	The identifier for a library of qualificatuin test components for a Python-based Command Line Interface.
tsTestsGUI	The identifier for a library of qualificatuin test components for a "wxPython"-style, "nCurses"-based Graphical-style User Interface.
tsToolsCLI	The identifier for a library of software development, diagnostic, maintenance and project management components for a Python-based Command Line Interface.
tsToolsGUI	The identifier for a library of software development, diagnostic, maintenance and project management components for a "wxPython"-style, "nCurses"-based Graphical-style User Interface.
tsUtilities	The identifier for a library of computer system configuration, diagnostic, installation, maintenance and performance tool components for various host hardware and software platforms.
tsWxGTUI	<p>The identifier for a library of building-block components for a "wxPython"-style, "nCurses"-based Graphical-style User Interface (tsToolkitGUI) and Python-based Command Line Interface (tsToolkitCLI).</p> <p>The tsToolkitGUI and tsToolkitCLI component organization keeps the Command Line Interface components independent of the Graphical-style User Interface ones. However, it does not preclude the Graphical-style User Interface components from using the services of the Command Line Interface components as appropriate.</p>
tsWxGTUI_PyVx	The generic identifier for the following Python language generation specific <i>TeamSTARS</i> "tsWxGTUI" Toolkit releases:

TERM	DEFINITION
	<ul style="list-style-type: none"> ▪ tsWxGTUI_Py2x --- Python 2.0.0 - 2.7.9 (Toolkit for 2nd generation Python language) ▪ tsWxGTUI_Py3x --- Python 3.0.0 - 3.4.2 (Toolkit for 3rd generation Python language)
tsWxGTUI_PyVx -Major .Minor .Maintenance	<p>The Major-Minor-Maintenance identifier for a <i>TeamSTARS</i> "tsWxGTUI" Toolkit release where:</p> <ul style="list-style-type: none"> ▪ Major --- A version number that denotes the introduction of a new design which cannot be expected to be backward compatible to a previous version. Zero (0) denotes the initial release for a product preview during its pre-alpha or beta stage. ▪ Minor --- A version number that denotes a product update that selectively introduces new features but otherwise can be expected to be backward compatible to a previous version. Zero (0) denotes the initial release. ▪ Maintenance --- A version number that denotes a product update that corrects defects found after the release of a previous version and otherwise can be expected to be backward compatible to that previous version. Zero (0) denotes a product release in its pre-alpha stage.
UNIX	<p>From Wikipedia, the free encyclopedia</p> <p>"Unix (officially trademarked as UNIX, sometimes also written as Unix) is a multitasking, multi-user computer operating system originally developed in 1969 by a group of AT&T employees at Bell Labs, including Ken Thompson, Dennis Ritchie, Brian Kernighan, Douglas McIlroy, Michael Lesk and Joe Ossanna. The Unix operating system was first developed in assembly language, but by 1973 had been almost entirely recoded in C, greatly facilitating its further development and porting to other hardware. Today's Unix system evolution is split into various branches, developed over time by AT&T as well as various commercial vendors, universities (such as University of California, Berkeley's BSD), and non-profit organizations."</p>
UWIN	<p>From Wikipedia, the free encyclopedia</p> <p>"UWIN is a computer software package created by David Korn which allows programs written for the operating system Unix be built and run on Microsoft Windows with few, if any, changes. Some of the software development was subcontracted to Wipro, India. References, correct or not, to the software as U/Win and AT&T Unix for Windows can be found in some cases, especially from the early days of its existence.</p> <p>UWIN source and binaries are available under the Open Source Eclipse Public License 1.0 at AT&T AST/UWIN open source downloads."</p> <p>See also:</p> <ul style="list-style-type: none"> ▪ Cygwin, a similar project started at about the same time[2] ▪ Interix, the Microsoft product in this area ▪ Windows Services for Unix ▪ MKS Toolkit, a third-party proprietary product in this area ▪ DJGPP (DJ's GNU Programming Platform), a Windows port of Gnu programming tools ▪ Xming ▪ UnxUtils ▪ GnuWin32 ▪ GNUWin II ▪ MinGW ▪ LBW: Linux Binaries on Windows requires Interix to be installed first."
VMware Fusion	<p>From Wikipedia, the free encyclopedia</p>

TERM	DEFINITION
	<p>"VMware Fusion is a software hypervisor developed by VMware for Macintosh computers with Intel processors. Fusion allows Intel-based Macs to run operating systems, such as Microsoft Windows, Linux, NetWare or Solaris on virtual machines, along with their Mac OS X operating system using a combination of paravirtualization, hardware virtualization and dynamic recompilation."</p>
VNC	<p>From Wikipedia, the free encyclopedia:</p> <p>"In computing, Virtual Network Computing (VNC) is a graphical desktop sharing system that uses the Remote Frame Buffer protocol (RFB) to remotely control another computer. It transmits the keyboard and mouse events from one computer to another, relaying the graphical screen updates back in the other direction, over a network.[1]</p> <p>VNC is platform-independent – a VNC viewer on one operating system may connect to a VNC server on the same or any other operating system. There are clients and servers for many GUI-based operating systems and for Java. Multiple clients may connect to a VNC server at the same time. Popular uses for this technology include remote technical support and accessing files on one's work computer from one's home computer, or vice versa.</p> <p>VNC was originally developed at the Olivetti & Oracle Research Lab in Cambridge, United Kingdom. The original VNC source code and many modern derivatives are open source under the GNU General Public License.</p> <p>VNC in KDE 3.1</p> <p>There are a number of variants of VNC[2] which offer their own particular functionality; e.g., some optimised for Microsoft Windows, or offering file transfer (not part of VNC proper), etc. Many are compatible (without their added features) with VNC proper in the sense that a viewer of one flavour can connect with a server of another; others are based on VNC code but not compatible with standard VNC.</p> <p>VNC and RFB are registered trademarks of RealVNC Ltd. in the U.S. and in other countries."</p>
vt100	<p>From Wikipedia, the free encyclopedia:</p> <p>"The VT100 is a video terminal that was made by Digital Equipment Corporation (DEC). Its detailed attributes became the de facto standard for terminal emulators to emulate.</p> <p>History</p> <p>It was introduced in August 1978, following its predecessor, the VT52, and communicated with its host system over serial lines using the ASCII character set and control sequences (a.k.a. escape sequences) standardized by ANSI. The VT100 was also the first Digital mass-market terminal to incorporate "graphic renditions" (blinking, bolding, reverse video, and underlining) as well as a selectable 80 or 132 column display. All setup of the VT100 was accomplished using interactive displays presented on the screen; the setup data was stored in non-volatile memory within the terminal. The VT100 also introduced an additional character set that allowed the drawing of on-screen forms.</p> <p>The control sequences used by the VT100 family are based on the ANSI X3.64 standard, also known as ECMA-48 and ISO/IEC 6429. These are sometimes referred to as ANSI escape codes. The VT100 was not the first terminal to be based on X3.64—The Heath Company had a microprocessor-based video terminal, the Heathkit H-19 (H19), that implemented a subset of the standard proposed by ANSI in X3.64.[1] In addition, the VT100 provided backwards compatibility for VT52 users, with support for the VT52 control sequences.[2]</p> <p>In 1983, the VT100 was replaced by the more-powerful VT200 series terminals such as the VT220. In August 1995 the terminal business of Digital was sold to Boundless Technologies.[3]"</p>

TERM	DEFINITION
vt220	<p>From Wikipedia, the free encyclopedia:</p> <p>"DEC VT220 terminal with LK201 keyboard.</p> <p>The VT220 was a terminal produced by Digital Equipment Corporation from 1983 to 1987.[1][2]</p> <p>Hardware</p> <p>The VT220 improved on the earlier VT100 series of terminals with a redesigned keyboard, much smaller physical packaging, and a much faster microprocessor. To meet the needs of various national regulatory agencies[citation needed], the VT220 was available with CRTs that used white, green, or amber phosphors.</p> <p>Several of the VT2xx models were pyramid shaped, allowing them to sit on a table top, leaving the surface of the display at an angle to the user; this angle could be adjusted. Because it was lower than head height, the result was an especially ergonomic terminal. The LK201 keyboard supplied with the VT220 was one of the first full length low profile keyboards available; it was developed at DEC's Roxbury, Massachusetts facility.</p> <p>The VT240 and VT241 were variants of the VT220, both capable of displaying vector graphics using the ReGIS instruction set. The VT241 was equipped with a color screen. The successor of the VT220 was the VT320, itself followed by the VT420.</p> <p>DEC VT220 connected to the serial port of a modern computer.</p> <p>Software</p> <p>The VT220 was designed to be compatible with the VT100, but added features to make it more suitable for an international market. This was accomplished with the National Replacement Character Set feature (e.g., Multinational Character Set) and support for 8-bit downloadable character sets."</p>
wxEmbedded	<p>From http://www.koansoftware.com/en/content/wxembedded:</p> <p>"wxEmbedded</p> <p>What is wxEmbedded®</p> <p>wxEmbedded® name and logo are registered trademarks of KOAN Software</p> <p>wxEmbedded - Embedded crossplatform GUI Library</p> <p>On March 2002 Koan software announced that a new project has been started by our company with wx-developers group.</p> <p>"After years of wishing, several recent projects have brought this closer to being a reality thanks to KOAN who has given the motivation behind all wxEmbedded project"</p> <p>-- Julian Smart, wxWidgets founder</p> <p>Here are the current strands in the wxEmbedded strategy, some points are already working, some are works in progress</p> <ul style="list-style-type: none"> ▪ wxWidgets for X11 (wxX11) ▪ wxWidgets for GTK+ (wxGTK) ▪ wxWidgets for Nano-X (wxNano-X) ▪ wxWidgets for Microwindows (wxMicrowindows) ▪ wxWidgets for SciTech MGL (wxMGL) ▪ wxWidgets for MS Windows CE (wxWinCE)

TERM	DEFINITION
	<ul style="list-style-type: none"> Host tools, such as wxEmulator Platforms supported are : x86 and ARM"
wxPython	A popular Python language binding for the cross-platform, "wxWidgets" GUI Toolkit.
wxWidgets	<p>A C++ library that lets developers create applications for Windows, OS X, Linux and UNIX on 32-bit and 64-bit architectures as well as several mobile platforms including Windows Mobile, iPhone SDK and embedded GTK+.</p> <p>It has popular language bindings for Python, Perl, Ruby and many other languages.</p> <p>"wxWidgets" (formerly "wxWindows") is a widget toolkit for creating graphical user interfaces (GUIs) for cross-platform applications. wxWidgets enables a program's GUI code to compile and run on several computer platforms with minimal or no code changes. It covers systems such as Microsoft Windows, Mac OS X (Carbon and Cocoa), iOS (Cocoa Touch), GNU/Linux Linux/Unix (X11, Motif, and GTK+), OpenVMS, OS/2 and AmigaOS. A version for embedded systems is under development.</p>
wxWindows License	<p>From Wikipedia, the free encyclopedia</p> <p>'wxWidgets is distributed under a custom made wxWindows License, similar to the GNU Lesser General Public License, with an exception stating that derived works in binary form may be distributed on the user's own terms.[5] This license is a free software license approved by the FSF,[17] making wxWidgets free software. It has been approved by the Open Source Initiative (OSI).[18]"</p>
xterm	<p>From Wikipedia, the free encyclopedia:</p> <p>"Not to be confused with X terminal display hardware.</p> <p>In computing, xterm is the standard terminal emulator for the X Window System. A user can have many different invocations of xterm running at once on the same display, each of which provides independent input/output for the process running in it (normally the process is a Unix shell).</p> <p>xterm originated prior to the X Window System. It was originally written as a stand-alone terminal emulator for the VAXStation 100 (VS100) by Mark Vandevoorde, a student of Jim Gettys, in the summer of 1984, when work on X started. It rapidly became clear that it would be more useful as part of X than as a standalone program, so it was retargeted to X. As Gettys tells the story, "part of why xterm's internals are so horrifying is that it was originally intended that a single process be able to drive multiple VS100 displays." [2]</p> <p>After many years as part of the X reference implementation, around 1996 the main line of development then shifted to XFree86 (which itself forked from X11R6.3), and it is presently actively maintained by Thomas Dickey.</p> <p>Many xterm variants are also available.[3] Most terminal emulators for X started as variations on xterm."</p> <p>NOTES:</p> <p>The "tsWxGTUI_PyVx" Toolkit supports the xterm, xterm-color, xterm-16color and xterm-256color terminal emulators with the following limitations:</p> <ul style="list-style-type: none"> The "tsWxGTUI_PyVx" Toolkit supports the xterm, xterm-color and xterm-16color terminal emulators. The inability to change the curses palette made it necessary to map the 68-color wxPython palette into the default 8-color curses palette. The "tsWxGTUI_PyVx" Toolkit does NOT yet support the xterm-256color terminal emulator. The inability to recreate the 68-color wxPython palette results in inappropriate colors and the

TERM	DEFINITION
	appearance of spurious lines streaking across the display.

7 APPENDIXES

Draft

Draft

8 APPENDIX A - BASELINE HOST COMPUTER PLATFORMS

This section shall identify those host computer platforms actually used to plan, implement, test, document, deploy and maintain the "tsWxGTUI_PyVx" Toolkit.

- 1 **Currently Used Platforms** (see "**Currently Used Platforms (Release Notes)**" on page 149) - Linux (Fedora 20 / OpenSUSE 13.1 / Scientific (CentOS) 6.5 / Ubuntu 12.04 & 14.04), Mac OS X (10.7 / 10.8 / 10.9), Microsoft Windows (8.1/8 Professional / 7 Professional / XP Professional each with Cygwin 1.7.x) and Unix (PC-BSD 10 & OpenIndiana 151a8 / Solaris 11 / SunOS 5.11).
 - a) 2013-model year, 3.5 GHz Intel Quad Core i7 processor-based desktop with 16 GB RAM and sufficient performance, resources and expansion capabilities to serve as the high-level baseline development and operator platform.
 - b) 2007-model year, 2.33 GHz Intel Core 2 Duo processor-based laptop with 4 GB RAM and sufficient performance, resources and expansion capabilities to serve as the moderate-level baseline development and operator platform.
 - c) 1999-model year, 366 MHz Intel Pentium II-based laptop with 384 MB RAM and expansion capabilities (marginal resources and performance) sufficient enough to serve as the low-level baseline development and operator platform.
- 2 **Previously Used Platforms** (see "**Previously Used Platforms (Release Notes)**" on page 156) - Linux (Fedora 15 & 16 / Open SuSE 11 / Ubuntu 10.04), Mac OS X (10.4 / 10.5 / 10.6 / 10.7), Microsoft Windows (8 Professional / 7 Professional / XP Professional each with Cygwin 1.7.x) and Unix (OpenIndiana 151a6 / Solaris 11 / SunOS 5.11).
 - a) 2007-model year, 2.33 GHz Intel Core 2 Duo processor-based laptop with 4 GB RAM and sufficient performance, resources and expansion capabilities to serve as the moderate-level baseline development and operator platform.
 - b) 1999-model year, 366 MHz Intel Pentium II-based laptop with 384 MB RAM and expansion capabilities (marginal resources and performance) sufficient enough to serve as the low-level baseline development and operator platform.
- 3 Designing Embedded Systems with Linux and Python - Link to YouTube Video by Mark Kohler.

8.1 Currently Used Platforms (Release Notes)

Host Computer Platform Configurations currently used by "tsWxGTUI_PyVx" Toolkit developers:

Draft

Make & Model	Hardware	Software
Apple 27" iMac Desktop	<p>2013-model year, 3.5 GHz Intel Quad Core i7 processor-based desktop with 16 GB RAM, 2560x1440 pixel resolution display and sufficient performance, resources and expansion capabilities to serve as the high-level baseline development and operator platform.</p> <ul style="list-style-type: none"> Its 3 TB (7200 RPM) SATA 6 Gb/s internal hard drive had 128 GB Solid State Flash memory and was used to run Apple's Mac OS X 10.9-10.10 and the hypervisor virtualization applications (Parallels Desktop 9-10 and VMware Fusion 5 and 7.1) that supported various guest operating systems. Its 3 TB (7200 RPM) SATA 6 Gb/s internal hard drive was also used to store and run configured versions of the eComStation Demo CD version 2.2 BETA (IBM OS/2 4.5 Warp descendant), CentOS Linux (7.0), Fedora Linux (21), OpenSUSE Linux (13.1), Scientific Linux (6.5), Ubuntu Linux (12.04, 14.04), Microsoft Windows (8.1 Pro, 8 Pro, 7 Pro, XP Pro and 2000 Pro), and Unix (PC-BSD (9.2, 10.0), OpenIndiana 151A8) guest operating systems. Hewlett-Packard Company P2015DN Series (26A5C8) LaserJet Printer connected via Ethernet Hewlett-Packard Company Officejet Pro 8500A (A910) e-All-in-One Series Printer connected via Wi-Fi or USB. 	<p>Host Operating System</p> <ul style="list-style-type: none"> Apple's Mac OS X (initially Mavericks releases 10.9.x and currently Yosemite releases 10.10.x) with Python 2.6.8, 2.7.5, 3.2.2 <p>Host Hypervisor Virtualization Applications</p> <ul style="list-style-type: none"> Parallels Desktop 9-10 VMware Fusion 5 & 7.1 <p>Concurrent or Interchangeable Guest Operating Systems (cloned from Apple 17" MacBook Pro Laptop and configured to share host drives and use 1440 x 900 display)</p> <ul style="list-style-type: none"> eComStation Demo CD version 2.2 BETA (IBM OS/2 4.5 Warp descendant) runs only what was shipped on the live CD and does not include Python. Most recent Python port for OS/2 & eComStations is Python 2.7.5. For details, see the following: "http://os2ports.smedley.id.au/index.php?page=python" and "http://blog.python.org/2011/05/python-33-to-drop-support-for-os2.html" Linux (Centos 7.0 64-bit, Fedora 21 64-bit, OpenSUSE 12.2 & 13.1 64-bit, Scientific 6.4 & 6.5 64-bit, Ubuntu 12.04 & 14.04 32-bit with Python 2.7 and 3.2. Microsoft Windows (8.1 Professional 32-bit, 8 Professional 32-bit, 7 Professional 32-bit with SP1, XP Professional 32-bit with SP3 and 2000 Professional 32-bit with SP2) with POSIX-like Cygwin plug-in and Python 2.6, 2.7, 3.2, 3.3 and 3.4. (NOTE: Windows 2000 came with obsolete Web Browser that precluded Windows 2000 updates and installation of Cygwin. After copying Windows 2000 updates and Cygwin directory from XP, Windows 2000 ran the TeamSTARS "tsWxGTUI_PyVx" Toolkit CLI and GUI tests but did not support mouse even with xterm.) UNIX (PC-BSD 9.2 & 10.0 64-bit without Parallels Tools, OpenIndiana 151a8 32-bit without Parallels Tools, a replacement for unreleased OpenSolaris 11/SunOS 5.11) with Python 2.6.4 running on Apple MacBook Pro
Apple 17"	2007-model year, 2.33 GHz Intel Core 2 Duo	Host Operating System

MacBook Pro Laptop	<p>processor-based laptop with 4 GB RAM, 1920x1200 pixel resolution display and sufficient performance, resources and expansion capabilities to serve as the moderate-level baseline development and operator platform.</p> <ul style="list-style-type: none"> ▪ Its 160 GB (5400 RPM) SATA 1.5 Gb/s internal hard drive was used to run Apple's Mac OS X 10.7 and the hypervisor virtualization applications (Parallels Desktop 8 and VMware Fusion 5) that supported various guest operating systems. ▪ Its 1.5 TB (7200 RPM) SATA 3 Gb/s external hard drive was used to store and run configured versions of the Ubuntu Linux (12.04), Microsoft Windows (8 Pro, 7 Pro and XP Pro) and Unix (OpenSolaris 11/OpenIndiana 151) guest operating systems. ▪ Hewlett-Packard Company P2015DN Series (26A5C8) LaserJet Printer connected via Ethernet ▪ Hewlett-Packard Company Officejet Pro 8500A (A910) e-All-in-One Series Printer connected via Wi-Fi or USB. 	<ul style="list-style-type: none"> ▪ Apple's Mac OS X 10.7.5 with Python 2.6.8, 2.7.5, 3.2.2 <p>Host Hypervisor Virtualization Applications</p> <ul style="list-style-type: none"> ▪ Parallels Desktop 8 ▪ VMware Fusion 5 <p>Interchangeable Guest Operating Systems (configured to share host drives and use 1440 x 900 display)</p> <ul style="list-style-type: none"> ▪ Linux (Fedora 20 64-bit, OpenSuSE 12.2 64-bit, Scientific (CentOS) 6.4-6.5 64-bit, Ubuntu 12.04 32-bit) with Python 2.7 and 3.2. ▪ Windows (8.1 Professional 32-bit, 8 Professional 32-bit, 7 Professional 32-bit with SP1, XP Professional 32-bit with SP3) with POSIX-like Cygwin plug-in and Python 2.6, 2.7, 3.2 and 3.3 ▪ UNIX (PC-BSD 9.2-10.0 64-bit without Parallels Tools, OpenIndiana 151A8 32-bit without Parallels Tools, a replacement for unreleased OpenSolaris 11/SunOS 5.11) with Python 2.6.4 running on Apple MacBook Pro
Dell 15.6" Inspiron 7000 Laptop	<p>1998-model year, 366 MHz Intel Pentium II-based laptop with 384 MB RAM, 640x480/1024x768 pixel resolution display and expansion capabilities (marginal resources and performance) sufficient enough to serve as the low-level baseline development and operator platform.</p> <ul style="list-style-type: none"> ▪ Its interchangeable 32 GB (4200 RPM) ATA hard drives were used to run Windows XP Pro or Ubuntu 12.04 Linux. ▪ The platform's limited memory and available PCMCIA network adapters were incompatible with later versions of Windows or with other Linux distributions. (Windows XP recognized Xircom Ethernet and 3Com Wireless adapters; Ubuntu Linux 12.04 recognized only Linksys Wireless adapter.) ▪ Hewlett-Packard Company Photosmart C3180 All-in-One Printer, Scanner, Copier connected via USB. 	<p>Interchangeable Host Operating System</p> <ul style="list-style-type: none"> ▪ Windows XP Professional with SP3) with POSIX-like Cygwin plug-in and Python 2.6, 2.7, 3.2 and 3.3 ▪ Linux (Ubuntu 12.04) with Python 2.7 and 3.3

NOTE: Since cross-platform operating system and Python virtual machine technology is also available for non-Intel based systems, it is likely that the TeamSTARS "tsWxGTUI_PyVx" toolkit will also work on those systems which use the equivalent operating systems and Python virtual machines with 32-bit and 64-bit microprocessors from other manufacturers including:

- *AMD*
- *ARM Holdings*
- *Cyrix*
- *Freescall*
- *Intel*
- *IBM*
- *Marvell*
- *NexGen*
- *Nvidia Tegra*
- *Oracle (previously Sun)*
- *OWC*
- *Qualcomm*
- *Rise Technology*
- *Samsung*
- *SigmaTel*
- *Texas Instruments*
- *Transmeta*
- *tilera*
- *Via (Centaur Technology division)*
- *winchip*

8.1.1 Python 2x Configuration Spreadsheet

OS Type	Distribution	Datapath	Release	Python 2.0	Python 2.4	Python 2.5	Python 2.6	Python 2.7
Linux-style Windows Add-on	Cygwin	32-bit	1.7.28					2.7.3
Linux	Fedora	64-bit	20 (Stuck in Auto Update 5/13/201 4)				2.6.9	2.7.5
Linux	Scientific	64-bit	6.50				2.6.6	
Linux	Ubuntu	32-bit	12.04	2.0.1 (CLI)	2.4.6 (CLI)	2.5.6 (CLI)	2.6.9 (CLI)	2.7.3
Unix	PC-BSD	32-bit	10.00					2.7.6
Unix	OpenIndiana / OpenSolaris / SunOS	32-bit	151.a8 / 11 / 5.11				2.6.4	

Unix	Mac OS X	64-bit	10.9.2					2.7.5
Windows	7	32-bit	CYGWIN_NT-6.1					2.7.3
Windows	8.1	32-bit	CYGWIN_NT-6.3					2.7.3
Windows	XP (End-of-life on 8 April 2014)	32-bit	CYGWIN_NT-5.1				2.6.8	2.7.3
Windows	XP (End-of-life on 8 April 2014)	32-bit	5.1.2600					2.7.1

Key:	Python x.y.z
x.y.z	Runs "tsWxGTUI_PyVx" Toolkit in both CLI and GUI Modes. It supports curses (it can "import curses") and "ncurses" (it can "import _curses"). It supports tsLibCLI, tsToolsCLI, tsUtilities, tsLibGUI and tsToolsGUI.
x.y.z (CLI)	Runs "tsWxGTUI_PyVx" Toolkit only in Command Line Interface Mode. It supports curses (it can "import curses") but does NOT support "ncurses" (it cannot "import _curses"). It supports tsLibCLI, tsToolsCLI and tsUtilities.
	Untested configuration due to the absence of an installable Python virtual machine. Though one may download and build a Python virtual machine from source code, it has been time consuming to resolve build problems. Some are yet to be resolved.

8.1.2 Python 3x Configuration Spreadsheet

OS Type	Distribution	Datapath	Release	Python 3.0	Python 3.1	Python 3.2	Python 3.3	Python 3.4
Linux-style Windows Add-on	Cygwin	32-bit	1.7.28			3.2.3		
Linux	Fedora	64-bit	20 (Stuck in Auto Update 5/13/2014)			3.2.5	3.3.2	
Linux	Scientific	64-bit	6.50			3.2.3	3.3.4	
Linux	Ubuntu	32-bit	12.04	3.0.1 (CLI)	3.1.5	3.2.3	3.3.4	3.4.0

Unix	PC-BSD	32-bit	10.00			3.2.3		
Unix	OpenIndiana / OpenSolaris / SunOS	32-bit	151.a8 / 11 / 5.11			3.2.5 (CLI)		
Unix	Mac OS X	64-bit	10.9.2			3.2.3	3.3.0	
Windows	7	32-bit	CYGWIN _NT-6.1			3.2.3		
Windows	8.1	32-bit	CYGWIN _NT-6.3			3.2.3		
Windows	XP (End-of- life on 8 April 2014)	32-bit	CYGWIN _NT-5.1			3.2.3		
Windows	XP (End-of- life on 8 April 2014)	32-bit	5.1.2600			3.2.3 (CLI)	3.3.1 (CLI)	3.4.0 (CLI)

Key:	Python x.y.z
x.y.z	Runs "tsWxGTUI_PyVx" Toolkit in both CLI and GUI Modes. It supports curses (it can "import curses") and "ncurses" (it can "import _curses"). It supports tsLibCLI, tsToolsCLI, tsUtilities, tsLibGUI and tsToolsGUI.
x.y.z (CLI)	Runs "tsWxGTUI_PyVx" Toolkit only in Command Line Interface Mode. It supports curses (it can "import curses") but does NOT support "ncurses" (it cannot "import _curses"). It supports tsLibCLI, tsToolsCLI and tsUtilities.
	Untested configuration due to the absence of an installable Python virtual machine. Though one may download and build a Python virtual machine from source code, it has been time consuming to resolve build problems. Some are yet to be resolved.

8.2 Previously Used Platforms (Release Notes)

Host Computer Platform Configurations previously used by "tsWxGTUI_PyVx" Toolkit developers:

Make & Model	Hardware	Software
Apple 17" MacBook Pro Laptop	<p>2007-model year, 2.33 GHz Intel Core 2 Duo processor-based laptop with 4 GB RAM and sufficient performance, resources and expansion capabilities to serve as the moderate-level baseline development and operator platform.</p> <ul style="list-style-type: none"> Its 160 GB (5400 RPM) SATA 1.5 Gb/s internal hard drive was used to run Apple's Mac OS X 10.7 and the hypervisor virtualization applications (Parallels Desktop 4-7 and VMware Fusion 4-5) that supported various guest operating systems. Its 1.5 TB (7200 RPM) SATA 3 Gb/s external hard drive was used to store and run configured versions of the Ubuntu Linux (10.04), Microsoft Windows (7 Pro and XP Pro) and Unix (OpenSolaris 11/OpenIndiana 151) guest operating systems. 	<p>Host Operating System</p> <ul style="list-style-type: none"> Apple's Mac OS X 10.4-10.7 with Python 2.6.8 <p>Host Hypervisor Virtualization Applications</p> <ul style="list-style-type: none"> Parallels Desktop 4-7 VMware Fusion 4-5 <p>Interchangeable Guest Operating Systems (configured to share host drives and use 1440 x 900 display)</p> <ul style="list-style-type: none"> Linux (Ubuntu 10.04) with Python 2.7 and 3.2 Windows (7 Professional / XP Professional with SP3) with POSIX-like Cygwin plug-in and Python 2.6, 2.7 and 3.2 UNIX (OpenIndiana 151a5, a replacement for unreleased OpenSolaris 11/SunOS 5.11) with Python 2.6.4 running on Apple MacBook Pro
Dell 15.6" Inspiron 7000 Laptop	<p>1998-model year, 366 MHz Intel Pentium II-based laptop with 384 MB RAM and expansion capabilities (marginal resources and performance) sufficient enough to serve as the low-level baseline development and operator platform.</p> <ul style="list-style-type: none"> Its interchangeable 32 GB (4200 RPM) ATA hard drives were used to run Windows XP Pro or Ubuntu 12.04 Linux. The platform's limited memory and available PCMCIA network adapters were incompatible with later versions of Windows or with other Linux distributions. (Windows XP recognized Xircom Ethernet and 3Com Wireless adapters; Ubuntu Linux 12.04 recognized only Linksys Wireless adapter.) 	<p>Interchangeable Host Operating System</p> <ul style="list-style-type: none"> Windows XP Professional with SP3) with POSIX-like Cygwin plug-in and Python 2.6, 2.7 and 3.2 Linux (Ubuntu 12.04) with Python 2.7 and 3.2

<p>Notes - Baseline Host Computer Platform Configurations previously used by "tsWxGTUI_PyVx" Toolkit developers</p>	<ol style="list-style-type: none"> 1 Linux (Fedora 15-16) with Python 2.6, 2.7 and 3.2 running on Apple MacBook Pro - 32-/64-bit Intel Core 2 Duo Processors running with 4 GB RAM Linux Virtual Machine created and managed by Parallels Desktop 6 for Mac 2 Linux (openSuSE 11) with Python 2.6, 2.7 and 3.2 running on Apple MacBook Pro - 32-/64-bit Intel Core 2 Duo Processors with 4 GB RAM running Virtual Machine created and managed by Parallels Desktop 6 for Mac 3 Linux (Ubuntu 10.04) with Python 2.7 and 3.2 running on Apple MacBook Pro - 32-/64-bit Intel Core 2 Duo Processors with 4 GB RAM running Linux Virtual Machine created and managed by Parallels Desktop 6 for Mac 4 Mac OS X (Tiger 10.4.0-Snow Leopard 10.6.8) with Python 2.6, 2.7 and 3.2 running on Apple MacBook Pro - 32-/64-bit Intel Core 2 Duo Processors with 4 GB RAM running Mac OS 10.4-10.6. 5 Windows (XP-SP3) with UNIX-like Cygwin plug-in and Python 2.6 running on Dell Laptop - 32-bit Intel Pentium 2 Processor with 384 MB RAM running Windows XP-SP3 6 Windows (XP-SP3 and Release 8 Preview) with UNIX-like Cygwin plug-in and Python 2.6 running on Apple MacBook Pro - 32-/64-bit Intel Core 2 Duo Processors with 4 GB RAM running Windows Virtual Machine created and managed by Parallels Desktop 6 for Mac
---	--

Notes - Experimental Host Computer Platform Configurations previously used by "tsWxGTUI_PyVx" Toolkit developers	<p>1 Windows (7 Professional) with built-in "Command Prompt" accessory shell and Python 2.7 running on Apple MacBook Pro - 32-/64-bit Intel Core 2 Duo Processors with 4 GB RAM running Windows Virtual Machine created and managed by Parallels Desktop (8) for Mac.</p> <hr/> <p>This configuration uses the Windows built-in "Command Prompt" accessory shell which can run Command Line Interface components ("tsLibCLI", "tsTestsCLI" and "tsToolsCLI") of the "tsWxGTUI_PyVx" toolkit.</p> <p>It does NOT support the low-level, "nCurses" library, a pre-requisite for running the Graphical-style User Interface components ("tsLibGUI", "tsTestsGUI" and "tsToolsGUI") of the "tsWxGTUI_PyVx" toolkit.</p>
	<p>2 Windows (7 Professional) with UNIX-like Git Bash plug-in and Python 2.7 running on Apple MacBook Pro - 32-/64-bit Intel Core 2 Duo Processors with 4 GB RAM running Windows Virtual Machine created and managed by Parallels Desktop (8) for Mac.</p> <hr/> <p>This configuration, like those using Cygwin, includes a Bash shell which can run Command Line Interface components ("tsLibCLI", "tsTestsCLI" and "tsToolsCLI") of the "tsWxGTUI_PyVx" toolkit.</p> <p>Unlike those configurations using Cygwin, it does NOT support the low-level, "nCurses" library, a pre-requisite for running the Graphical-style User Interface components ("tsLibGUI", "tsTestsGUI" and "tsToolsGUI") of the "tsWxGTUI_PyVx" toolkit.</p>
	<p>3 Windows (7 Professional) and Python 2.7 with the GNUwin32 and PDCurses Version 2.6 plug-ins running on Apple MacBook Pro - 32-/64-bit Intel Core 2 Duo Processors with 4 GB RAM running Windows Virtual Machine created and managed by Parallels Desktop (8) for Mac.</p> <hr/> <p>This configuration, unlike those using Cygwin, includes a DOS-like Command Prompt shell which can run Command Line Interface components ("tsLibCLI", "tsTestsCLI" and "tsToolsCLI") of the "tsWxGTUI_PyVx" toolkit.</p> <p>Like those configurations using Cygwin, PDCurses Version 2.6 does support the low-level, Python "Curses" library, a pre-requisite for running the Graphical-style User Interface components ("tsLibGUI", "tsTestsGUI" and "tsToolsGUI") of the "tsWxGTUI_PyVx" toolkit. It runs the test_PDCurses (renamed test_tsWxCurses) application. However, PDCurses Version 2.6 traps because it lacks the mouse button definitions needed by the "tsWxGraphicalTextUserInterface" module to emulate "wxPython" in order to run such applications as "test_tsWxWidgets.py".</p>

9 APPENDIX B - API RELATIONSHIP

An Application Programming Interface (API) is a source code based specification intended to be used as an interface by software components to communicate with each other. An API may include specifications for routines, data structures, object classes, and variables.

- 1 High Level API** - A programming interface that is the least detailed but provides more functionality within one statement than a lower-level interface. High-level interfaces are designed to enable the programmer to write code in a shorter amount of time and to be less involved with the details of the software module or hardware device that is performing the required services. For example, the programmer can invoke a high level procedure to append one or more text strings to an internal buffer.
- 2 Low Level API** - A programming interface that is the most detailed, allowing the programmer to manipulate functions within a software module or within hardware at a very granular level. To continue with the afore mentioned example, a high level procedure can then invoke one or more low level procedures that will sequentially get one text string at a time from the internal buffer, sequentially scroll the top most string off the display, then scroll up each remaining displayed string and finally outputting the new string to the bottom row of the screen.
- 3 Extended API** - A customized version of an existing programming interface that supports additional keyword value pairs and positional arguments. For example, the "tsWxGTUI_PyVx" Toolkit is a character-mode emulation of the pixel-mode "wxPython" API. It internally may require optional parameters to distinguish such features as character-mode dimensions from their pixel-mode counterparts. It may also include functionality that "wxPython" and its underlying "wxWidgets" toolkit otherwise obtain from the host operating system and its programming libraries, such as the installation of the color palette and the implementation of scrollable windows.

9.1 Comparison of wxPython with tsWxGTUI (Development Plan)

This tabulation shall briefly compare the features, capabilities and limitations of the pixel-mode "wxPython" / "wxWidgets" / "wxEmbedded" Toolkit with those of its character-mode emulator, the "tsWxGTUI_PyVx" Toolkit.

Feature	"wxPython" / "wxWidgets" / "wxEmbedded"	"tsWxGTUI_PyVx"
Platform	Locally (via system console or xterm) and remotely (via vnc) accessible systems: <ul style="list-style-type: none"> ▪ Desktop ▪ Laptop ▪ Workstation 	Locally (via system console or xterm) and remotely (via xterm) accessible systems: <ul style="list-style-type: none"> ▪ Desktop ▪ Laptop ▪ Workstation

Feature	"wxPython" / "wxWidgets" / "wxEmbedded"	"tsWxGTUI_PyVx"
	<ul style="list-style-type: none"> ▪ Embedded (via KOAN's "wxEmbedded") 	<ul style="list-style-type: none"> ▪ Embedded
Operator Desktop Interface Hardware	<ul style="list-style-type: none"> ▪ Keyboard ▪ Mouse, trackball, touchpad or touchscreen ▪ Optional Mouseless ▪ Display (pixel mode) 	<ul style="list-style-type: none"> ▪ Keyboard ▪ Mouse, trackball, touchpad or touchscreen ▪ Keyboard Shortcut Key with/without Optional Mouse (Future) ▪ Display (character mode)
Operating System & Device Driver Software	<ul style="list-style-type: none"> ▪ Linux ▪ Mac OS X ▪ Microsoft Windows ▪ Unix 	<ul style="list-style-type: none"> ▪ Linux ▪ Mac OS X ▪ Microsoft Windows (with free add-on of POSIX-like Cygwin Command Line Interface and GNU tools from Red Hat) ▪ Unix
Terminal Device Interface Software	<ul style="list-style-type: none"> ▪ "wxWidgets" internally uses host Operating System specific API for its Terminal Device Interface. ▪ It translates host Operating System specific events into "wxWidget" specific published API events. ▪ Events include keyboard key press / release, mouse button press / release and single / double click with mouse position at every clocked sample. ▪ Events also include window resizing. 	<ul style="list-style-type: none"> ▪ "nCurses" internally uses host Operating System specific API for its Terminal Device Interface. ▪ It translates host Operating System specific events into "nCurses" specific published API events. ▪ Events include keyboard key press / release, mouse button press / release and single / double / triple click with mouse position ONLY available at time of button state change. ▪ Events also include window resizing. However, event handling is currently limited to trapping the unsupported event. Though re-initializing "nCurses" to detect and use the new size is feasible and fast (50 milliseconds or more depending on host processor and terminal color palette (and associated definition of or mapping of wxPython color palette), it does not address the time consuming (20 seconds or more on host processor) complexities associated with re-initializing the "wxPython" emulation and the System Operator designated application program.
Programming Language	<ul style="list-style-type: none"> ▪ "wxWidgets" is implemented in C++ ▪ "wxPython" binding/wrapper for "wxWidgets" is implemented with SWIG in Python 2.x and Python 3.x 	<ul style="list-style-type: none"> ▪ "tsWxGTUI_PyVx" is implemented in Python 2.x ▪ After debugging, "tsWxGTUI_PyVx" is ported from Python 2.x to Python 3.x via the standard Python "2to3" utility with minor manual debugging when appropriate
Terminal Interface Software	<ul style="list-style-type: none"> ▪ "wxWidgets" internally uses Operating System or X window system specific API for Graphical User Interface. 	<ul style="list-style-type: none"> ▪ "tsWxGTUI_PyVx" internally uses Python Curses ("nCurses") module API for Graphical-style User Interface. The Python Curses module only implements the most

Feature	"wxPython" / "wxWidgets" / "wxEmbedded"	"tsWxGTUI_PyVx"
		<p>commonly used subset features of the "nCurses" API.</p> <ul style="list-style-type: none"> ▪ The "tsWxGTUI_PyVx" Toolkit emulates a typical Operating System or X window system specific API for Graphical User Interface.
Operator Desktop Interface Software	<ul style="list-style-type: none"> ▪ Host Operating System specific Command Line Interface is a POSIX compatible shell (such as bash) ▪ Host Operating System specific Graphical User Interface features support multiple independently re-sizable and repositionable Frames and Dialogs ▪ Host Operating System specific Graphical User Interface features reflect proprietary placement of labels and buttons to iconize, maximize/restore and close frames and dialogs ▪ Host Operating System specific Graphical User Interface task bar features support the closing of individual Frames and Dialogs ▪ Host Operating System specific task bar features support the shifting of foreground focus from one Frame or Dialog to another 	<ul style="list-style-type: none"> ▪ Host Operating System specific Command Line Interface is a POSIX compatible shell (such as bash) ▪ Host Operating System specific Graphical User Interface shell features support multiple independently re-sizable and repositionable Command Line Interface shell windows ▪ "tsWxGTUI_PyVx" emulated Graphical User Interface features DO NOT support multiple independently re-sizable and repositionable Frames and Dialogs WITHOUT the System Operator first creating separate Command Line Interface shell windows ▪ "tsWxGTUI_PyVx" emulated Graphical User Interface features reflect Microsoft Windows-style placement of labels and buttons to iconize, maximize/restore and close frames and dialogs ▪ "tsWxGTUI_PyVx" emulated Graphical User Interface task bar features DO NOT currently support the closing of individual Frames and Dialogs ▪ "tsWxGTUI_PyVx" Toolkit task bar features (future enhancement) support the shifting of foreground focus from one Frame or Dialog to another
Application Programming Interface	<ul style="list-style-type: none"> ▪ "wxWidgets" / "wxPython" API for Graphical User Interface supports bit-mapped images. ▪ It supports fixed and variable sized fonts and at least the 68 most commonly used colors. 	<ul style="list-style-type: none"> ▪ "tsWxGTUI_PyVx" emulated subset of "wxWidgets" / "wxPython" API for Graphical-style User Interface DOES NOT support bitmapped images except for the single, predefined bitmapped image used as a splash screen at startup ▪ It supports the 1-color (single terminal-dependant green, orange or white phosphor) that is "ON" or "OFF" (black) associated with a non-color, VT100/VT220 terminal hardware and terminal emulator software. ▪ It supports a single fixed sized font and at least the 68 most commonly used colors (which are either internally mapped into the "nCurses" standard 8-color, 16-color or defined for optional 88-color and 256-color

Feature	"wxPython" / "wxWidgets" / "wxEmbedded"	"tsWxGTUI_PyVx"
		<p>terminal hardware and terminal emulator software).</p> <ul style="list-style-type: none">▪ It supports a 71-color palette by augmenting the 68-color wxPython palette with the 3 colors that must be supported for the xterm-16color palette.▪ It supports a 140-color palette by augmenting the 68-color wxPython palette with the 3 colors unique to the xterm-16color palette plus 69 other colors that demonstrate the customization potential. However, the constraint on the number of color pairs (32767) constrains the number of colors to be no more than 181 (square root of 32768) unless the design is re-designed to use a sparse matrix to avoid impractical color combinations.

Draft

10 APPENDIX C - DELIVERABLES

The following table describes the work product(s) to be delivered by the developer:

DELIVERABLE	DESCRIPTION
Software Package(s)	<p>The "tsWxGTUI_PyVx" Toolkit software:</p> <ul style="list-style-type: none"> ▪ Building Block Library Packages ("tsLibCLI" & "tsLibGUI") in Python 2x & Python 3x source code formats ▪ Tool Application Programs ("tsToolsCLI" & "tsToolsGUI") in Python 2x & Python 3x source code formats ▪ Test Application Programs ("tsTestsCLI" & "tsTestsGUI") in Python 2x & Python 3x source code formats ▪ Utility Modules ("tsUtilities") in Python 2x and Python 3x and POSIX-style shell script formats
Engineering Documentation ("tsEngineering") (Optional, Fee-based subscription)	<p>While the <i>TeamSTARS</i> "tsWxGTUI_PyVx" Toolkit is released as free, open source software, the engineering documentation establishes proprietary Copyright ownership by the original Toolkit Author(s).</p> <p>Toolkit recipients can obtain a license for the engineering documentation, if their intent is to assemble a large team to commercialize the software and want to kickstart the effort.</p> <p>The "tsEngineering" subdirectory contains a collection of Toolit Author written large, complex documents, including structured documents, featuring multiple fonts and graphic images.</p> <p>The documents are authored using the following products:</p> <ul style="list-style-type: none"> ▪ Author-it --- A help authoring tool (http://www.author-it.com) and content management system for creating, maintaining, and distributing single-sourced content. It generates Microsoft Word files with the appropriate outline numbering (for table of contents, sections, paragraphs, lists and figures) regardless of where the single-source content originated. ▪ Microsoft Office --- A collection of software applications (http://www.microsoftstore.com/store/msusa/en_US/cat/Office/categoryID.62684700) intended to be used by knowledge workers. The components are generally distributed together, have a consistent user interface and usually can interact with each other, sometimes in ways that the operating system would not normally allow. <p>Access (data base)</p> <p>Excel (spreadsheet)</p> <p>PowerPoint (presentation)</p> <p>Word (word processor)</p>

DELIVERABLE	DESCRIPTION
Software Package(s)	<p>The "tsWxGTUI_PyVx" Toolkit software:</p> <ul style="list-style-type: none"> ▪ Building Block Library Packages ("tsLibCLI" & "tsLibGUI") in Python 2x & Python 3x source code formats ▪ Tool Application Programs ("tsToolsCLI" & "tsToolsGUI") in Python 2x & Python 3x source code formats ▪ Test Application Programs ("tsTestsCLI" & "tsTestsGUI") in Python 2x & Python 3x source code formats ▪ Utility Modules ("tsUtilities") in Python 2x and Python 3x and POSIX-style shell script formats
	<ul style="list-style-type: none"> ▪ Microsoft Visio --- A software application (http://www.microsoftstore.com/store/msusa/en_US/list/Visio/categoryID.62687700) intended to be used by knowledge workers. Visio (diagrams) ▪ Fineprint Software (http://fineprint.com) FinePrint (A print driver, for Microsoft Windows, which allows you to manage the printing process. It helps you to reduce printing costs while enhancing and managing complex print jobs.) pdfFactory Pro (An application, for Microsoft Windows, that provides Adobe PDF compatible output.) <p>The documents accompany the computer software for the training and reference use of the software developer:</p> <ul style="list-style-type: none"> ▪ It explains the purpose, goals, non-goals, system requirements, interface requirements, software requirements, qualification requirements, development plan, software design, how it operates and how to install, use and troubleshoot it. ▪ It is provided in various application specific formats (such as Adobe PDF and Microsoft Office & Visio formats which may be read and edited by the free, open source, cross-platform LibreOffice Suite). <p>The "tsWxGTUI_PyVx" Toolkit software development and release documents are deliverable in Adobe's Portable Document Format and Microsoft's Word Format (with associated bit-mapped images in JPG or PNG Format):</p> <ul style="list-style-type: none"> ▪ Vol. 0 --- SDIST Announcement ▪ Vol. 1 --- SDIST Brochure ▪ Vol. 2 --- SDIST Introduction ▪ Vol. 3 --- SDIST Terms & Conditions ▪ Vol. 4 --- SDIST Software Development Plans ▪ Vol. 5 --- SDIST System Specification ▪ Vol. 6 --- SDIST Interface Requirements Specification ▪ Vol. 7 --- SDIST Software Requirements Specification ▪ Vol. 8 --- SDIST Release Notes ▪ Vol. 9 --- SDIST Software User's Manual

DELIVERABLE	DESCRIPTION
Software Package(s)	<p>The "tsWxGTUI_PyVx" Toolkit software:</p> <ul style="list-style-type: none"> ▪ Building Block Library Packages ("tsLibCLI" & "tsLibGUI") in Python 2x & Python 3x source code formats ▪ Tool Application Programs ("tsToolsCLI" & "tsToolsGUI") in Python 2x & Python 3x source code formats ▪ Test Application Programs ("tsTestsCLI" & "tsTestsGUI") in Python 2x & Python 3x source code formats ▪ Utility Modules ("tsUtilities") in Python 2x and Python 3x and POSIX-style shell script formats
	<ul style="list-style-type: none"> ▪ Vol. 10 --- SDIST Appendixes ▪ Vol. 11 --- SDIST Dictionary ▪ Vol. 12 --- SDIST To-Do List
Engineering Notes	<p>The "tsWxGTUI_PyVx" Toolkit software development notes in Microsoft's Word Format:</p> <ul style="list-style-type: none"> ▪ Class List.doc ▪ Relationships_Between_tsWxGTUI_Toolkit_APIs.doc ▪ RGB to Color Name Mapping(Triplet and Hex).doc ▪ Writing a Python Package by Tarek Ziadé.doc <p>The "tsWxGTUI_PyVx" Toolkit software development notes in Microsoft's Access Format:</p> <ul style="list-style-type: none"> ▪ tsWxPython.mdb <p>The "tsWxGTUI_PyVx" Toolkit software development diagram notes in Microsoft's Visio Format:</p> <ul style="list-style-type: none"> ▪ Distributed System.vsd ▪ Networked Architecture.vsd ▪ System Architecture.vsd ▪ tsWxPython Org Chart.vsd <p>The "tsWxGTUI_PyVx" Toolkit software development notes in Microsoft's Excel Format:</p> <ul style="list-style-type: none"> ▪ Platform_Configuration.xls <p>The "tsWxGTUI_PyVx" Toolkit software development notes in Plain Text Format:</p> <ul style="list-style-type: none"> ▪ README_BMP.txt
Equipment Operator Documentation ("tsDocuments")	<p>The "tsWxGTUI_PyVx" Toolkit installation, configuration, operation and troubleshooting documents in Plain Text Format:</p> <ul style="list-style-type: none"> ▪ tsDocCLI-1-GettingStartedFiles ▪ tsDocCLI-2-DistributionReadMeFiles ▪ tsDocCLI-3-DeveloperReadMeFiles ▪ tsDocCLI-5-UsageTermsAndConditions ▪ tsManPages ("tsLibCLI", "tsLibGUI", "tsToolsCLI", "tsToolsGUI", "tsTestsCLI", "tsTestsGUI") <p>1. Operator Documentation</p> <p>User documentation for each "tsWxGTUI" Toolkit distribution is contained, in plain text</p>

DELIVERABLE	DESCRIPTION
Software Package(s)	<p>The "tsWxGTUI_PyVx" Toolkit software:</p> <ul style="list-style-type: none"> ▪ Building Block Library Packages ("tsLibCLI" & "tsLibGUI") in Python 2x & Python 3x source code formats ▪ Tool Application Programs ("tsToolsCLI" & "tsToolsGUI") in Python 2x & Python 3x source code formats ▪ Test Application Programs ("tsTestsCLI" & "tsTestsGUI") in Python 2x & Python 3x source code formats ▪ Utility Modules ("tsUtilities") in Python 2x and Python 3x and POSIX-style shell script formats
	<p>format, in the subdirectory named ".tsDocCLI". The subdirectory is organized, by topic, to contain the following:</p> <p>a. How to Prepare Platform to Get Started</p> <p>The "./GettingStartedFiles" subdirectory contains the following file(s):</p> <p>"README-GettingStarted.txt" ---</p> <p>b. How to Install and Redistribute</p> <p>The "./DistributionReadMeFiles" subdirectory contains the following file(s):</p> <ul style="list-style-type: none"> ▪ "AUTHORS.txt" --- List of the principal "tsWxGTUI" Toolkit author(s) and authors credited for work covered by a prior copyright and license. ▪ "BUGS.txt" --- List of Known Problems / Issues. ▪ "CHANGE_LOG.txt" --- List of Additions, Modification and Deletions. ▪ "CONFIGURE.txt" --- Instructions for applying factory and site-specific configurations. ▪ "COPYING.txt" --- Instructions for copying all or a portion of the distribution. ▪ "FAQ.txt" --- Answers to Frequently Asked Questions. ▪ "INSTALL.txt" --- Describes steps to download, extract install and configure the "tsWxGUI" Toolkit. ▪ "LICENSE.txt" --- General and special arrangements, provisions, rules, specifications and standards that form an integral part the agreement or contract between the creator and recipient of Copyrighted and Licensed Work. ▪ "MANIFEST.txt" --- Tally List for deliverable items. ▪ "NEWS.txt" --- Announcements of new releases. ▪ "NOTICES.txt" --- Details the copyright(s) and license(s). ▪ "OPERATE.txt" --- Describes steps to use the "tsWxGUI" Toolkit. ▪ "README.txt" --- Introduces new recipients to the purpose, goals, non-goals, design and features of the computer software product. ▪ "THANKS.txt" --- Acknowledgments to those otherwise unsung heros who contributed time and effort to supporting the authors as planners, editors, designers, coders and testers. ▪ "TO-DO.txt" --- A To-Do-List provides a roadmap for development and troubleshooting work. ▪ "TROUBLE SHOOT.txt" --- Provides a list of available reference resources and a guide for planning, developing and troubleshooting a cross-platform system of

DELIVERABLE	DESCRIPTION
Software Package(s)	<p>The "tsWxGTUI_PyVx" Toolkit software:</p> <ul style="list-style-type: none"> ▪ Building Block Library Packages ("tsLibCLI" & "tsLibGUI") in Python 2x & Python 3x source code formats ▪ Tool Application Programs ("tsToolsCLI" & "tsToolsGUI") in Python 2x & Python 3x source code formats ▪ Test Application Programs ("tsTestsCLI" & "tsTestsGUI") in Python 2x & Python 3x source code formats ▪ Utility Modules ("tsUtilities") in Python 2x and Python 3x and POSIX-style shell script formats
	<p>hundreds of files each containing a few, tens or hundred of class, data and method definitions. Its complexity becomes apparent in the recent software Lines-Of-Code metrics.</p> <p>c. How to Use and Modify</p> <p>The "./DeveloperReadMeFiles" subdirectory contains the following file(s):</p> <ul style="list-style-type: none"> ▪ "README.txt" ▪ "README_1st.txt" ▪ "README_1st-PyPI-Dev-tsWxGTUI.txt" ▪ "README-01-Title_Page.txt" ▪ "README-02-Table_Of_Contents.txt" ▪ "README-03-Purpose.txt" ▪ "README-04-Goals.txt" ▪ "README-05-Non-Goals.txt" ▪ "README-06-Design_Strategy.txt" ▪ "README-07-Design_Architecture.txt" ▪ "README-08-Software_Configuration_Management.txt" ▪ "README-09-tsWxGTUI_Directories.txt" ▪ "README-10-tsToolkitCLI_Directories.txt" ▪ "README-11-tsToolkitGUI_Directories.txt" ▪ "README-12-Features.txt" ▪ "README-13-Current_Capabilities.txt" ▪ "README-14-Current_Limitations.txt" ▪ "README-15-Reference_Documents.txt" <p>d. How to Learn "tsWxGTUI" Toolkit Software Development</p> <p>The "./DeveloperTutorialFiles" subdirectory contains the following file(s):</p> <ul style="list-style-type: none"> ▪ "CLI_0_hello_world_print_statement.py" ▪ "CLI_1_hello_world_print_function.py" ▪ "CLI_2_hello_world_script_environment.py" ▪ "CLI_3_hello_world_main_module_application.py" ▪ "GUI_4_Curses_LowLevel_WidgetApi_application.py"

DELIVERABLE	DESCRIPTION
Software Package(s)	<p>The "tsWxGTUI_PyVx" Toolkit software:</p> <ul style="list-style-type: none"> ▪ Building Block Library Packages ("tsLibCLI" & "tsLibGUI") in Python 2x & Python 3x source code formats ▪ Tool Application Programs ("tsToolsCLI" & "tsToolsGUI") in Python 2x & Python 3x source code formats ▪ Test Application Programs ("tsTestsCLI" & "tsTestsGUI") in Python 2x & Python 3x source code formats ▪ Utility Modules ("tsUtilities") in Python 2x and Python 3x and POSIX-style shell script formats
	<ul style="list-style-type: none"> ▪ "GUI_5_tsWxGTUI_HighLevel_WidgetApi_application.py" ▪ "GUI_6_tsWxGTUI_HighLevel_BoxSizerApi_application.py" ▪ "ReadMe_Developer_Tutorial_Files.txt" ▪ "test_tsCxGlobals.py" ▪ "test_tsWxGlobals.py" ▪ "tsCxGlobals.py" <p>e. Terms & Conditions</p> <p>The "./UsageTermsAndConditions" subdirectory contains the following file(s):</p> <ul style="list-style-type: none"> ▪ "COPYRIGHT.txt" ▪ "LICENSE.txt" ▪ "NOTICES.txt" ▪ "SplashScreenDesignersGuide.txt"

11 APPENDIX D - LOG FILES

11.1 Log File Examples

- *Bit-Mapped Image Tree (Splash Screen Files for Graphical Applications)* (on page 170)
- *Logs Tree (Files for Non-Graphical Applications)* (on page 172)
- *Logs Tree (Files for Graphical Applications)* (on page 182)

Draft

11.1.1 Bit-Mapped Image Tree (Splash Screen Files for Graphical Applications)

```
# File: ".logs/bmp/README_BMP.txt"
# "Time-stamp: <01/05/2015  2:14:20 AM rsg>"
```

This "bmp" directory contains those Splash Screen(s) generated by:

```
"/tsWxGTUI_Py2x/tsLibGUI/tsWxPkg/src/tsWxGraphicalTextUserInterface.py.
```

A Splash Screen is displayed during the launch of a GUI-style application program. It identifies the application's author(s), copyright(s) and licence(s) using information defined in:

```
"/tsWxGTUI_Py2x/tsLibGUI/tsWxPkg/src/tsWxGlobals.py"
```

Splash Screens are named for the display size (in character columns and rows/lines), terminal/emulator type, and host operating system.

Examples:

Basic Multi-Color ("wxPython" transformation maps
68-color palette into 8 or 16 built-in colors)

Base Name	Size	Type	Host OS	File Ext.	Notes
"SplashScreen-[60x15_CYGWIN]-cygwin_nt-5.0.bmp"					(Placeholder for Windows 2000)
"SplashScreen-[60x15_CYGWIN]-cygwin_nt-5.1.bmp"					(Windows XP)
"SplashScreen-[60x15_CYGWIN]-cygwin_nt-5.2.bmp"					(Placeholder for Windows XP x64)
"SplashScreen-[60x15_CYGWIN]-cygwin_nt-6.0.bmp"					(Placeholder for Windows Vista)
"SplashScreen-[60x15_CYGWIN]-cygwin_nt-6.1.bmp"					(Windows 7)
"SplashScreen-[60x15_LINUX]-cygwin_nt-6.1.bmp"					(Windows 7)
"SplashScreen-[60x15_XTERM]-cygwin_nt-6.1.bmp"					(Windows 7)
"SplashScreen-[80x25_XTERM]-darwin.bmp"					(Mac OS 10.9.1)
"SplashScreen-[96x40_XTERM]-linux.bmp"					(Fedora 21)
"SplashScreen-[96x40_XTERM]-linux.bmp"					(Ubuntu 12.04)
"SplashScreen-[128x50_XTERM]-freebsd.bmp"					(PC-BSD 9.2)
"SplashScreen-[128x50_XTERM]-sunos.bmp"					(OpenIndiana 151a8)
"SplashScreen-[60x15_XTERM-COLOR]-cygwin_nt-6.1.bmp"					(Windows 7)
"SplashScreen-[80x15_XTERM-16COLOR]-cygwin_nt-6.3.bmp"					(Windows 8.1)
"SplashScreen-[80x15_XTERM-88COLOR]-cygwin_nt-6.3.bmp"					(16x16 color pair limit for Windows 8.1)
"SplashScreen-[80x15_XTERM-256COLOR]-cygwin_nt-6.3.bmp"					(16x16 color pair limit for Windows 8.1)
"SplashScreen-[80x15_XTERM-256COLOR]-cygwin_nt-6.4.bmp"					(16x16 color pair limit for Windows 10)

Non-Color ("wxPython" transformation maps 68-color palette

into black with one shade of the default color for displays having only a white, orange or green phosphor).

Base Name	Size	Type	Host OS	File Ext.	Notes
"SplashScreen-[80x40_VT100]-cygwin_nt-5.0.bmp"					(Placeholder for Windows 2000)
"SplashScreen-[80x40_VT100]-cygwin_nt-5.1.bmp"					(Windows XP)
"SplashScreen-[80x40_VT100]-cygwin_nt-5.2.bmp"					(Placeholder for Windows XP x64)
"SplashScreen-[80x40_VT100]-cygwin_nt-6.0.bmp"					(Placeholder for Windows Vista)
"SplashScreen-[80x15_VT100]-cygwin_nt-6.1.bmp"					(Windows 7)
"SplashScreen-[80x15_VT100]-cygwin_nt-6.2.bmp"					(Windows 8)
"SplashScreen-[80x15_VT220]-cygwin_nt-6.3.bmp"					(Windows 8.1)
"SplashScreen-[80x15_VT220]-cygwin_nt-6.4.bmp"					(Windows 10)

```
#####
# Advanced Multi-Color support is still evolving.... #
# # #
# Terminal/Emulator standards support an undocumented #
# maximum of 256 color pairs. This inference resulted #
# from the following observations. #
# # #
# Under Python 2.x and Python 3.x, most xterm-88color #
# and xterm-256color terminal emulators produced the #
# wrong colors marred by spurious underline artifacts. #
# # #
# Under Python 3.3.0 with a Yosemite Mac OS X Terminal #
# utility, there were 256 color pairs and a 16-color #
# palette that worked. #
# # #
# Also, under the Python 2.7.6 with the GNOME Desktop #
# for CentOS 7.0 Linux, there were 256 color pairs and #
# a 16-color palette that worked . #
# # #
# For the color pair matrix used by the "tsWxGTUI" #
# Toolkit, the Advanced Multi-Color support emulates #
# xterm-16color and associated mapping of the wxPython #
# 68-color palette into the built-in 16 colors. #
#####
```

```
if (COLOR_PAIRS > 256) and (USE_256_COLOR_PAIR_LIMIT): # As set in
tsWxGlobals.py
```

```
Advanced Multi-Color ("wxPython" emulation maps
68-color palette into 16 of 88 or 16 of 256 colors)
```

```
elif (COLOR_PAIRS == 256) # As reported by curses
```

```
Advanced Multi-Color ("wxPython" emulation maps
68-color palette into 16 of 88 or 16 of 256 colors)
```

```
else:
```

```
Advanced Multi-Color ("wxPython" emulation maps
```

68-color palette into 71 of 88 or 140 of 256 colors)

Base Name	Size	Type	Host OS	File Ext.	Notes
"SplashScreen-[80x15_XTERM-88COLOR]-cygwin_nt-6.3.bmp"					(Windows 8.1)
"SplashScreen-[80x25_XTERM-88COLOR]-darwin.bmp"					(Mac OS 10.9.1)
"SplashScreen-[96x40_XTERM-88COLOR]-linux.bmp"					(Fedora 20)
"SplashScreen-[80x15_XTERM-256COLOR]-cygwin_nt-6.3.bmp"					(Windows 8.1)
"SplashScreen-[80x25_XTERM-256COLOR]-darwin.bmp"					(Mac OS 10.9.1)
"SplashScreen-[96x40_XTERM-256COLOR]-linux.bmp"					(Fedora 20)

A new Splash Screen is built upon the first use of a uniquely sized command line interface display by those host operating systems that share this directory.

NOTE:

Previous terminal emulator used in a command line interface shell can alter the built-in color palette for subsequent terminal emulators.
Examples:

- 1) The final xterm sees no color palette change if the first one is xterm followed by xterm-color, vt100 and xterm.
- 2) The final xterm sees a dim color palette change if the first one is xterm followed by xterm-16color, vt100 and xterm.

Keeping a copy here avoids spending time to rebuild the same Splash Screen each time a GUI-style application program is launched.

You may recover disk space by deleting those Splash Screens that have outlived their usefulness.

11.1.2 **Logs Tree (Files for Non-Graphical Applications)**

The "tsWxGTUI_PyVx" Toolkit registers, via log files, the startup, configuration and chronology of activities needed for design verification and troubleshooting.

- 1** The root directory is named "logs" and is always placed in the directory from which the application has been launched. The root directory is created, if it does not already exist.

- a) Within the "logs" directory is a date and time stamped directory for each application startup. The startup directory is always created.
- b) Within the startup directory are the application-specific log files.

"PlatformRunTimeEnvironment.log" - Captures current hardware, software and network information about the run time environment for the user process.

"<application>.log(s)" - Captures date and time stamped messages associated with application designated normal and abnormal situations and activities. The contents and verbosity of "<application>.log(s)" are controlled by configuration settings (for details see Customizable CLI Features).

- 2 It is left to the System Administrator, Software Engineer, System Operator and Field Service personnel to do the housekeeping that removes those log file when they are no longer useful.

```
/cygdrive/d/WR/SoftwareGadgetry-Dev/Python-2x/tsWxGTUI/logs
rsg@RICHARDSGOR69ED /cygdrive/d/WR/SoftwareGadgetry-Dev/Python-2x/tsWxGTUI/logs
$ tree ./
./
├── 2013-12-06-at-03-29-09
│   ├── PlatformRunTimeEnvironment.log
│   ├── Redirected-stdout.log
│   ├── TerminalRunTimeEnvironment.log
│   └── test_tsWxWidgets.log
├── 2013-12-06-at-03-38-05
│   ├── PlatformRunTimeEnvironment.log
│   ├── Redirected-stdout.log
│   ├── TerminalRunTimeEnvironment.log
│   └── test_tsWxWidgets.log
├── 2013-12-06-at-03-44-55
│   ├── PlatformRunTimeEnvironment.log
│   ├── TerminalRunTimeEnvironment.log
│   └── test_tsWxWidgets.log
└── 2013-12-06-at-03-50-56
    ├── displayDictionaryTest.log
    ├── PlatformRunTimeEnvironment.log
    └── test_tsReportUtilities.log

4 directories, 14 files

rsg@RICHARDSGOR69ED /cygdrive/d/WR/SoftwareGadgetry-Dev/Python-2x/tsWxGTUI/logs
$
```

11.1.2.1 2013-12-06-at-03-50-56

11.1.2.1.1 PlatformRunTimeEnvironment.log

tsPlatformRunTimeEnvironment.py, v2.1.0 (build 10/19/2013)

Author(s): Richard S. Gordon
Copyright (c) 2007-2013 Richard S. Gordon.
All rights reserved.
GNU General Public License, Version 3, 29 June 2007

Credits:

tsLibCLI Import & Application Launch Features:
Copyright (c) 2007-2009 Frederick A. Kier.
All rights reserved.
GNU General Public License, Version 3, 29 June 2007

terminalsize (<https://gist.github.com/jtriley/1108174>) Features:
Copyright (c) 2011 Justin T. Riley.
All rights reserved.
GNU General Public License, Version 3, 29 June 2007

Python Platform & Logging Module API Features:
Copyright (c) 2001-2013 Python Software Foundation.
All rights reserved.
PSF License Agreement for Python 2.7.3 & 3.3.0

===== Begin Platform Run Time Environment =====

Reported Fri, 06-Dec-2013 at 03:50:58

Network Identification

hostname = <RICHARDSGOR69ED>
aliaslist = <[]> # NOTE: List of Values NOT available.
ipaddrlist = <['fe80::5079:ab53:c361:23a4']>

Host Central Processing Unit

machine = <i686>
processor = <> # NOTE: Value NOT available.
architecture = <32> bits; <> # NOTE: Value NOT available. linkage
byteorder = <little>

Host Operating System

api = <posix>
system = <CYGWIN_NT-6.1>


```

release = <1.7.25(0.270/5/3)>
version = <2013-08-31 20:39>

```

Host Console Display Size

```

-----
characters/line = <80>
lines/display = <35>

```

Python Platform

```

-----
branch = <> # NOTE: Value NOT available.
build = <default> number; <Dec 18 2012 13:50:09> date
compiler = <GCC 4.5.3>
implementation = <CPython>
revision = <> # NOTE: Value NOT available.
version = <2.7.3>

```

Process Parameters

```

-----
pid = <900496> / <0xDBD90>
getppid = <891332> / <0xD99C4>
getegid = <513>
geteuid = <1000>
getgid = <513>
getgroups = <[545, 1001, 513]>
getpgid = <900496>
getuid = <1000>
getlogin = <rsg>
ctermid = </dev/pty0>
cwd = </cygdrive/d/WR/SoftwareGadgetry-Dev/Python-2x/tsWxGTUI>

```

Environment Variables

```

-----
!:: = <::\>
ALLUSERSPROFILE = <C:\ProgramData>
APPDATA = <C:\Users\rsg\AppData\Roaming>
COMMONPROGRAMFILES = <C:\Program Files\Common Files>
COMPUTERNAME = <RICHARDSGOR69ED>
COMSPEC = <C:\Windows\system32\cmd.exe>
EMAIL = <D:\Mail\The Bat!>
FP_NO_HOST_CHECK = <NO>
HOME = </home/rsg>
HOMEDRIVE = <d:>
HOMEPATH = <\Home\rsg>
HOSTNAME = <RICHARDSGOR69ED>
INFOPATH = </usr/local/info:/usr/share/info:/usr/info:>
LANG = <en_US.UTF-8>
LOCALAPPDATA = <C:\Users\rsg\AppData\Local>
LOGONSERVER = <\\RICHARDSGOR69ED>
MANPATH =
    </usr/local/man:/usr/share/man:/usr/man::/usr/ssl
    /man>

```

```

NUMBER_OF_PROCESSORS = <1>
    OLDPWD = </home/rsg>
    OS = <Windows_NT>
    PATH = </usr/local/bin:/usr/bin:/cygdrive/c/Program
        Files/Parallels/Parallels Tools/Applications:/cyg
        drive/c/Windows/system32:/cygdrive/c/Windows:/cyg
        drive/c/Windows/System32/Wbem:/cygdrive/c/Windows
        /System32/WindowsPowerShell/v1.0:/cygdrive/c/PROG
        RA~1/CONDUS~1/DISKEE~1:/cygdrive/c/Program
        Files/Microsoft SQL
        Server/110/Tools/Binn:/cygdrive/c/Program
        Files/Microsoft SQL
        Server/110/DTS/Binn:/cygdrive/c/Program
        Files/Microsoft SQL Server/110/Tools/Binn/Managem
        entStudio:/cygdrive/c/Borland/Bcc55/Bin:/cygdrive
        /c/Python27:/cygdrive/c/Program Files/TortoiseHg:
        /cygdrive/c/gnuwin32/bin:/cygdrive/c/Bazaar:/cygd
        rive/c/Program Files/Mercurial:/usr/lib/lapack>
    PATHEXT =
        <.COM;.EXE;.BAT;.CMD;.VBS;.VBE;.JS;.JSE;.WSF;.WSH
        ;.MSC>
    PRINTER = <HP LaserJet P2015 Series (26A5C8)>
PROCESSOR_ARCHITECTURE = <x86>
PROCESSOR_IDENTIFIER = <x86 Family 6 Model 60 Stepping 3, GenuineIntel>
PROCESSOR_LEVEL = <6>
PROCESSOR_REVISION = <3c03>
PROGRAMFILES = <C:\Program Files>
    PS1 = <[\e]0;\w\a\]\n\[\e[32m\]\u@\h
        \[\e[33m\]\w\[\e[0m\]\n$ >
PSModulePath =
    <C:\Windows\system32\WindowsPowerShell\v1.0\Modules\
    es;c:\Program Files\Microsoft SQL
    Server\110\Tools\PowerShell\Modules\>
PUBLIC = <C:\Users\Public>
PWD = </cygdrive/d/WR/SoftwareGadgetry-Dev/Python-
    2x/tsWxGTUI>
ProgramData = <C:\ProgramData>
SESSIONNAME = <Console>
SHELL = </bin/bash>
SHLVL = <1>
SYSTEMDRIVE = <C:>
SYSTEMROOT = <C:\Windows>
TERM = <xterm>
TZ = <America/New_York>
USER = <rsg>
USERDOMAIN = <RICHARDSGOR69ED>
USERNAME = <rsg>
USERPROFILE = <C:\Users\rsg>
WINDIR = <C:\Windows>
_ = </usr/bin/python>
temp = <C:\Users\rsg\AppData\Local\Temp>
tmp = <C:\Users\rsg\AppData\Local\Temp>
windows_tracing_flags = <3>
windows_tracing_logfile = <C:\BVTBin\Tests\installpackage\csilogfile.log>

```

===== End Platform Run Time Environment =====

Draft

11.1.2.1.2 test_tsReportUtilities.log

```
2013/12/06 03:50:58,881      INFO: tsApplication.TsApplication:
    "logs": <"[]">
2013/12/06 03:50:58,881      INFO: tsApplication.TsApplication:
    "Command Line": <"['test_tsReportUtilities.py']">
2013/12/06 03:50:58,881      INFO: tsApplication.TsApplication:
    "buildTitle" = <"test_tsReportUtilities.py">.
2013/12/06 03:50:58,881      INFO: tsApplication.TsApplication:
    "buildVersion" = <"2.0.0">.
2013/12/06 03:50:58,881      INFO: tsApplication.TsApplication:
    "buildDate" = <"05/24/2013">.
2013/12/06 03:50:58,881      INFO: tsApplication.TsApplication:
    "buildAuthors" = <"Richard S. Gordon">.
2013/12/06 03:50:58,881      INFO: tsApplication.TsApplication:
    "buildCopyright" = <"Copyright (c) 2007-2013 Richard S. Gordon.
    All rights reserved.">.
2013/12/06 03:50:58,881      INFO: tsApplication.TsApplication:
    "buildLicense" = <"GNU General Public License, Version 3, 29 June
2007">.
2013/12/06 03:50:58,881      INFO: tsApplication.TsApplication:
    "buildCredits" = <"

Credits:

    tsLibCLI Import & Application Launch Features:
    Copyright (c) 2007-2009 Frederick A. Kier.
    All rights reserved.">.
2013/12/06 03:50:58,881      INFO: tsApplication.TsApplication:
    "buildTitleVersionDate" = <"test_tsReportUtilities.py, v2.0.0 (build
05/24/2013)">.
2013/12/06 03:50:58,881      INFO: tsApplication.TsApplication:
    "buildHeader" = <"

test_tsReportUtilities.py, v2.0.0 (build 05/24/2013)

Author(s): Richard S. Gordon
Copyright (c) 2007-2013 Richard S. Gordon.
    All rights reserved.
GNU General Public License, Version 3, 29 June 2007

Credits:

    tsLibCLI Import & Application Launch Features:
    Copyright (c) 2007-2009 Frederick A. Kier.
    All rights reserved.
">.
2013/12/06 03:50:58,881      INFO: tsApplication.TsApplication:
    "purpose" not defined, using <"">
2013/12/06 03:50:58,881      INFO: tsApplication.TsApplication:
    "runTimeEntryPoint" = <"<function theMainApplication at 0x7fd4179c>">.
2013/12/06 03:50:58,881      INFO: tsApplication.TsApplication: "guiRequired"
not found, using <"False">
2013/12/06 03:50:58,881      INFO: tsApplication.TsApplication:
    "buildTitleVersionDate" = <"test_tsReportUtilities.py, v2.0.0 (build
05/24/2013)">.
```

```

2013/12/06 03:50:58,881      INFO: tsApplication.TsApplication:
    "buildHeader" = <

test_tsReportUtilities.py, v2.0.0 (build 05/24/2013)

Author(s): Richard S. Gordon
Copyright (c) 2007-2013 Richard S. Gordon.
    All rights reserved.
GNU General Public License, Version 3, 29 June 2007

Credits:

    tsLibCLI Import & Application Launch Features:
    Copyright (c) 2007-2009 Frederick A. Kier.
    All rights reserved.
">.
2013/12/06 03:50:58,881      INFO: tsApplication.TsApplication:
    "runTimeEntryPoint" = <"<function theMainApplication at 0x7fd4179c>">.
2013/12/06 03:50:58,881      INFO:

test_tsReportUtilities.py, v2.0.0 (build 05/24/2013)

Author(s): Richard S. Gordon
Copyright (c) 2007-2013 Richard S. Gordon.
    All rights reserved.
GNU General Public License, Version 3, 29 June 2007

Credits:

    tsLibCLI Import & Application Launch Features:
    Copyright (c) 2007-2009 Frederick A. Kier.
    All rights reserved.

2013/12/06 03:50:58,881      INFO:
*** DISPLAY DICTIONARY TEST ***

2013/12/06 03:50:58,885      INFO:
*** GET NEXT PATH NAME TEST ***

2013/12/06 03:50:58,885      INFO:      ./ junk ./junk_0001.txt
2013/12/06 03:50:58,885      INFO:
*** GET STATISTICS TIME TEST ***

2013/12/06 03:50:58,885      INFO:      1386319858.89 1386323458.89
    Started 03:50:58
    Ended   04:50:58
    Elapsed 00-01:00:00 (days-hrs:min:sec)

2013/12/06 03:50:58,885      INFO:
*** GET STATISTICS LIST TEST ***

2013/12/06 03:50:58,885      INFO:      1386319858.89 1386323458.89 100 80 20
Test Summary: FAILED after 00-01:00:00 (days-hrs:min:sec)
    Details: Runs 100, Passes 80, Failures 20
    Started 03:50:58
    Ended   04:50:58

```

Elapsed 00-01:00:00 (days-hrs:min:sec)

2013/12/06 03:50:58,885 INFO:

*** GET DAY HOUR MINUTE SECOND STRING TEST ***

2013/12/06 03:50:58,885 INFO: 0 00-00:00:00

2013/12/06 03:50:58,885 INFO: 1 00-00:00:01

2013/12/06 03:50:58,885 INFO: 59 00-00:00:59

2013/12/06 03:50:58,885 INFO: 60 00-00:01:00

2013/12/06 03:50:58,885 INFO: 120 00-00:02:00

2013/12/06 03:50:58,885 INFO: 3599 00-00:59:59

2013/12/06 03:50:58,885 INFO: 3600 00-01:00:00

2013/12/06 03:50:58,885 INFO: 86400 01-00:00:00

2013/12/06 03:50:58,885 INFO: 90000 01-01:00:00

2013/12/06 03:50:58,885 INFO:

*** GET SECONDS TIME FROM HOURS MINUTES SECONDS STRING TEST ***

2013/12/06 03:50:58,885 INFO: None 0

2013/12/06 03:50:58,885 INFO: 00:00:00 0

2013/12/06 03:50:58,885 INFO: 01:02:03 3723

2013/12/06 03:50:58,885 INFO: 12:34:56 45296

2013/12/06 03:50:58,885 INFO: 23:59:59 86399

2013/12/06 03:50:58,885 INFO: 24:00:00 86400

2013/12/06 03:50:58,885 INFO:

*** GET BYTE COUNT STRINGS TEST ***

2013/12/06 03:50:58,885 INFO: 1024^0 ('1 Bytes', '1', '0x1')

2013/12/06 03:50:58,885 INFO: 1024^1 ('1.00 KBytes', '1024', '0x400')

2013/12/06 03:50:58,885 INFO: 1024^2 ('1.00 MBytes', '1048576',
'0x100000')

2013/12/06 03:50:58,885 INFO: 1024^3 ('1.00 GBytes', '1073741824',
'0x40000000')

2013/12/06 03:50:58,885 INFO: 1024^4 ('1.00 TBytes', '1099511627776',
'0x10000000000')

2013/12/06 03:50:58,885 INFO: 1024^5 ('1.00 PBytes',
'1125899906842624', '0x40000000000000')

2013/12/06 03:50:58,885 INFO: 1024^6 ('1.00 EBytes',
'1152921504606846976', '0x1000000000000000')

2013/12/06 03:50:58,885 INFO: 1024^7 ('1.00 ZBytes',
'1180591620717411303424', '0x400000000000000000')

2013/12/06 03:50:58,885 INFO: 1024^8 ('1.00 YBytes',
'1208925819614629174706176', '0x10000000000000000000')

2013/12/06 03:50:58,885 INFO:

*** DISPLAY DICTIONARY TEST ***

11.1.2.1.3 displayDictionaryText.log

```

----- Begin "myDictionary" at level 0 -----

        ----- Begin "contents" at level 1 -----

                name = contents

                ----- Begin "programDictionary" at level 2 -----

                        list2 = ['ab', 'cd', 'ef']
                        main = __main__
                        mainTitleVersionDate = test_tsReportUtilities.py, v2.0.0
(build 05/24/2013)
                        name = programDictionary

                ----- End   "programDictionary" at level 2 -----

                ----- Begin "releaseDictionary" at level 2 -----

                        date = 05/24/2013
                        list1 = [12, 34, 56]
                        name = releaseDictionary
                        title = test_tsReportUtilities.py
                        version = 2.0.0

                ----- End   "releaseDictionary" at level 2 -----

                ----- Begin "sequentialDictionary" at level 2 -----

                        -1 = pear
                        15 = 1.234
                        25 = 1234 (0x4D2)
                        50 = orange
                        100 = apple
                        name = sequentialDictionary

                ----- End   "sequentialDictionary" at level 2 -----

        ----- End   "contents" at level 1 -----

        name = myDictionary

----- End   "myDictionary" at level 0 -----

```

11.1.3 Logs Tree (Files for Graphical Applications)

The "tsWxGTUI_PyVx" Toolkit registers, via log files, the startup, configuration and chronology of activities needed for design verification and troubleshooting.

- 1 The root directory is named "logs" and is always placed in the directory from which the application has been launched. The root directory is created, if it does not already exist.

- a) Within the "logs" directory is a date and time stamped directory for each application startup. The startup directory is always created.

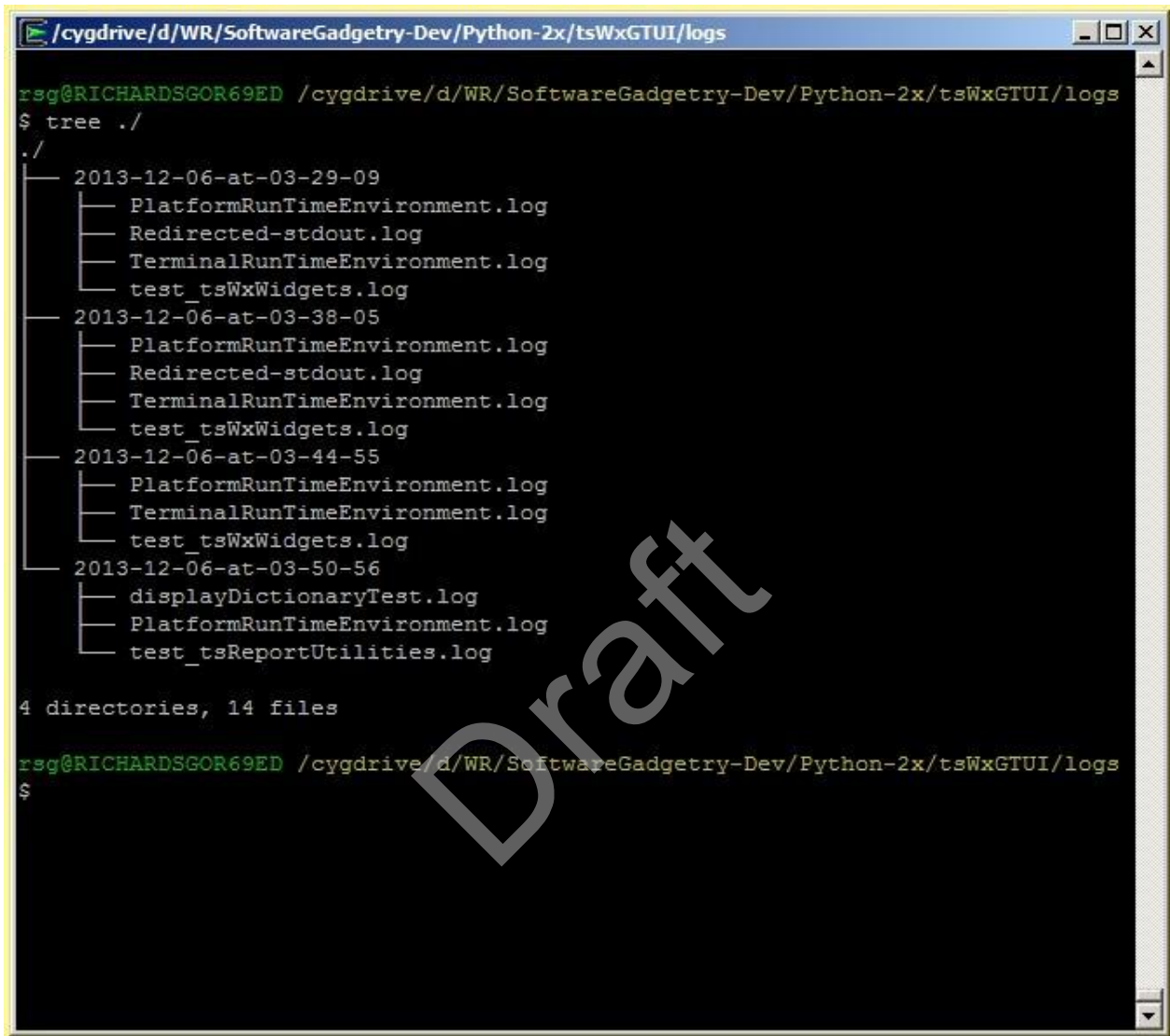
- b) Within the startup directory are the application-specific log files.

"PlatformRunTimeEnvironment.log" - Captures current hardware, software and network information about the run time environment for the user process.

"TerminalRunTimeEnvironment.log" - Captures the current hardware, software and emulated configuration of the "wxPython"-style, "nCurses"-based Graphical-Text User Interface subsystem. The contents and verbosity of "TerminalRunTimeEnvironment.log(s)" are controlled by configuration settings (for details see Customizable CLI Features).

"<application>.log(s)" - Captures date and time stamped messages associated with application designated normal and abnormal situations and activities. The contents and verbosity of "<application>.log(s)" are controlled by configuration settings (for details see Customizable CLI Features).

- 2 It is left to the System Administrator, Software Engineer, System Operator and Field Service personnel to do the housekeeping that removes those log file when they are no longer useful.



```
/cygdrive/d/WR/SoftwareGadgetry-Dev/Python-2x/tsWxGTUI/logs
rsg@RICHARDSGOR69ED /cygdrive/d/WR/SoftwareGadgetry-Dev/Python-2x/tsWxGTUI/logs
$ tree ./
./
├── 2013-12-06-at-03-29-09
│   ├── PlatformRunTimeEnvironment.log
│   ├── Redirected-stdout.log
│   ├── TerminalRunTimeEnvironment.log
│   └── test_tsWxWidgets.log
├── 2013-12-06-at-03-38-05
│   ├── PlatformRunTimeEnvironment.log
│   ├── Redirected-stdout.log
│   ├── TerminalRunTimeEnvironment.log
│   └── test_tsWxWidgets.log
├── 2013-12-06-at-03-44-55
│   ├── PlatformRunTimeEnvironment.log
│   ├── TerminalRunTimeEnvironment.log
│   └── test_tsWxWidgets.log
└── 2013-12-06-at-03-50-56
    ├── displayDictionaryTest.log
    ├── PlatformRunTimeEnvironment.log
    └── test_tsReportUtilities.log

4 directories, 14 files
rsg@RICHARDSGOR69ED /cygdrive/d/WR/SoftwareGadgetry-Dev/Python-2x/tsWxGTUI/logs
$
```

11.1.3.1 2013-12-06-at-03-44-55

11.1.3.1.1 PlatformRunTimeEnvironment.log

tsPlatformRunTimeEnvironment.py, v2.1.0 (build 10/19/2013)

Author(s): Richard S. Gordon
Copyright (c) 2007-2013 Richard S. Gordon.
All rights reserved.
GNU General Public License, Version 3, 29 June 2007

Credits:

tsLibCLI Import & Application Launch Features:
Copyright (c) 2007-2009 Frederick A. Kier.
All rights reserved.
GNU General Public License, Version 3, 29 June 2007

terminalsize (<https://gist.github.com/jtriley/1108174>) Features:
Copyright (c) 2011 Justin T. Riley.
All rights reserved.
GNU General Public License, Version 3, 29 June 2007

Python Platform & Logging Module API Features:
Copyright (c) 2001-2013 Python Software Foundation.
All rights reserved.
PSF License Agreement for Python 2.7.3 & 3.3.0

===== Begin Platform Run Time Environment =====

Reported Fri, 06-Dec-2013 at 03:44:57

Network Identification

hostname = <RICHARDSGOR69ED>
aliaslist = <[]> # NOTE: List of Values NOT available.
ipaddrlist = <['fe80::5079:ab53:c361:23a4']>

Host Central Processing Unit

machine = <i686>
processor = <> # NOTE: Value NOT available.
architecture = <32> bits; <> # NOTE: Value NOT available. linkage
byteorder = <little>

Host Operating System

api = <posix>
system = <CYGWIN_NT-6.1>

```
release = <1.7.25(0.270/5/3)>
version = <2013-08-31 20:39>
```

Host Console Display Size

```
-----
characters/line = <80>
lines/display = <35>
```

Python Platform

```
-----
branch = <> # NOTE: Value NOT available.
build = <default> number; <Dec 18 2012 13:50:09> date
compiler = <GCC 4.5.3>
implementation = <CPython>
revision = <> # NOTE: Value NOT available.
version = <2.7.3>
```

Process Parameters

```
-----
pid = <898588> / <0xDB61C>
getppid = <891332> / <0xD99C4>
getegid = <513>
geteuid = <1000>
getgid = <513>
getgroups = <[545, 1001, 513]>
getpgid = <898588>
getuid = <1000>
getlogin = <rsg>
ctermid = </dev/pty0>
cwd = </cygdrive/d/WR/SoftwareGadgetry-Dev/Python-2x/tsWxGTUI>
```

Environment Variables

```
-----
!:: = <::\>
ALLUSERSPROFILE = <C:\ProgramData>
APPDATA = <C:\Users\rsg\AppData\Roaming>
COMMONPROGRAMFILES = <C:\Program Files\Common Files>
COMPUTERNAME = <RICHARDSGOR69ED>
COMSPEC = <C:\Windows\system32\cmd.exe>
EMAIL = <D:\Mail\The Bat!>
FP_NO_HOST_CHECK = <NO>
HOME = </home/rsg>
HOMEDRIVE = <d:>
HOMEPATH = <\Home\rsg>
HOSTNAME = <RICHARDSGOR69ED>
INFOPATH = </usr/local/info:/usr/share/info:/usr/info:>
LANG = <en_US.UTF-8>
LOCALAPPDATA = <C:\Users\rsg\AppData\Local>
LOGONSERVER = <\\RICHARDSGOR69ED>
MANPATH =
    </usr/local/man:/usr/share/man:/usr/man::/usr/ssl
    /man>
```

```

NUMBER_OF_PROCESSORS = <1>
    OLDPWD = </home/rsg>
    OS = <Windows_NT>
    PATH = </usr/local/bin:/usr/bin:/cygdrive/c/Program
        Files/Parallels/Parallels Tools/Applications:/cyg
        drive/c/Windows/system32:/cygdrive/c/Windows:/cyg
        drive/c/Windows/System32/Wbem:/cygdrive/c/Windows
        /System32/WindowsPowerShell/v1.0:/cygdrive/c/PROG
        RA~1/CONDUS~1/DISKEE~1:/cygdrive/c/Program
        Files/Microsoft SQL
        Server/110/Tools/Binn:/cygdrive/c/Program
        Files/Microsoft SQL
        Server/110/DTS/Binn:/cygdrive/c/Program
        Files/Microsoft SQL Server/110/Tools/Binn/Managem
        entStudio:/cygdrive/c/Borland/Bcc55/Bin:/cygdrive
        /c/Python27:/cygdrive/c/Program Files/TortoiseHg:
        /cygdrive/c/gnuwin32/bin:/cygdrive/c/Bazaar:/cygd
        rive/c/Program Files/Mercurial:/usr/lib/lapack>
    PATHEXT =
        <.COM;.EXE;.BAT;.CMD;.VBS;.VBE;.JS;.JSE;.WSF;.WSH
        ;.MSC>
    PRINTER = <HP LaserJet P2015 Series (26A5C8)>
PROCESSOR_ARCHITECTURE = <x86>
PROCESSOR_IDENTIFIER = <x86 Family 6 Model 60 Stepping 3, GenuineIntel>
PROCESSOR_LEVEL = <6>
PROCESSOR_REVISION = <3c03>
PROGRAMFILES = <C:\Program Files>
    PS1 = <\[e]0;\w\a\]\n\[e[32m]\u@h
        \[e[33m]\w\[e[0m]\n$ >
PSModulePath =
    <C:\Windows\system32\WindowsPowerShell\v1.0\Modules\
    ;c:\Program Files\Microsoft SQL
    Server\110\Tools\PowerShell\Modules\>
PUBLIC = <C:\Users\Public>
PWD = </cygdrive/d/WR/SoftwareGadgetry-Dev/Python-
    2x/tsWxGTUI>
ProgramData = <C:\ProgramData>
SESSIONNAME = <Console>
SHELL = </bin/bash>
SHLVL = <1>
SYSTEMDRIVE = <C:>
SYSTEMROOT = <C:\Windows>
TERM = <xterm>
TZ = <America/New_York>
USER = <rsg>
USERDOMAIN = <RICHARDSGOR69ED>
USERNAME = <rsg>
USERPROFILE = <C:\Users\rsg>
WINDIR = <C:\Windows>
_ = </usr/bin/python>
temp = <C:\Users\rsg\AppData\Local\Temp>
tmp = <C:\Users\rsg\AppData\Local\Temp>
windows_tracing_flags = <3>
windows_tracing_logfile = <C:\BVTBin\Tests\installpackage\csilogfile.log>

```

===== End Platform Run Time Environment =====

Draft

11.1.3.1.2 test_tsWxWidgets.log

```
2013/12/06 03:44:58,407      INFO: tsApplication.TsApplication:
    "logs": <"['']">
2013/12/06 03:44:58,407      INFO: tsApplication.TsApplication:
    "Command Line": <"['test_tsWxWidgets.py']">
2013/12/06 03:44:58,407      INFO: tsApplication.TsApplication:
    "buildTitle" = <"test_tsWxWidgets">.
2013/12/06 03:44:58,407      INFO: tsApplication.TsApplication:
    "buildVersion" = <"2.3.0">.
2013/12/06 03:44:58,407      INFO: tsApplication.TsApplication:
    "buildDate" = <"06/04/2013">.
2013/12/06 03:44:58,407      INFO: tsApplication.TsApplication:
    "buildAuthors" = <"Richard S. Gordon">.
2013/12/06 03:44:58,407      INFO: tsApplication.TsApplication:
    "buildCopyright" = <"Copyright (c) 2007-2013 Richard S. Gordon.
    All rights reserved.">.
2013/12/06 03:44:58,407      INFO: tsApplication.TsApplication:
    "buildLicense" = <"GNU General Public License, Version 3, 29 June
2007">.
2013/12/06 03:44:58,407      INFO: tsApplication.TsApplication:
    "buildCredits" = <"
```

Credits:

```
    tsLibGUI Import & Application Launch Features:
    Copyright (c) 2007-2009 Frederick A. Kier.
    All rights reserved.
```

```
    Python Curses Module API & Run Time Library Features:
    Copyright (c) 2001-2013 Python Software Foundation.
    All rights reserved.
```

```
    PSF License Agreement for Python 2.7.3 & 3.3.0
```

```
    wxWidgets (formerly wxWindows) & wxPython API Features:
    Copyright (c) 1992-2008 Julian Smart, Robert Roebling,
    Vadim Zeitlin and other members of the
    wxWidgets team.
    All rights reserved.
```

```
    wxWindows Library License
```

```
    nCurses API & Run Time Library Features:
    Copyright (c) 1998-2011 Free Software Foundation, Inc.
    All rights reserved.
```

```
    GNU General Public License, Version 3, 29 June 2007">.
```

```
2013/12/06 03:44:58,407      INFO: tsApplication.TsApplication:
    "buildTitleVersionDate" = <"test_tsWxWidgets, v2.3.0 (build
06/04/2013)">.
2013/12/06 03:44:58,407      INFO: tsApplication.TsApplication:
    "buildHeader" = <"
```

test_tsWxWidgets, v2.3.0 (build 06/04/2013)

```
    Author(s): Richard S. Gordon
    Copyright (c) 2007-2013 Richard S. Gordon.
    All rights reserved.
```

GNU General Public License, Version 3, 29 June 2007

Credits:

tsLibGUI Import & Application Launch Features:
Copyright (c) 2007-2009 Frederick A. Kier.
All rights reserved.

Python Curses Module API & Run Time Library Features:
Copyright (c) 2001-2013 Python Software Foundation.
All rights reserved.
PSF License Agreement for Python 2.7.3 & 3.3.0

wxWidgets (formerly wxWindows) & wxPython API Features:
Copyright (c) 1992-2008 Julian Smart, Robert Roebling,
Vadim Zeitlin and other members of the
wxWidgets team.
All rights reserved.
wxWindows Library License

nCurses API & Run Time Library Features:
Copyright (c) 1998-2011 Free Software Foundation, Inc.
All rights reserved.
GNU General Public License, Version 3, 29 June 2007

```
">.
2013/12/06 03:44:58,407      INFO: tsApplication.TsApplication:
      "buildPurpose" = <"
test_tsWxWidgets.py - Test application program. It demonstrates
features and operation of the tsWxGTUI toolkit and associated
building block components of tsLibCLI and tsLibGUI.
">.
2013/12/06 03:44:58,407      INFO: tsApplication.TsApplication:
      "runTimeEntryPoint" = <"<function EntryPoint at 0x7fcf464c>">.
2013/12/06 03:44:58,407      INFO: tsApplication.TsApplication:
      "guiRequired" = <"True">.
2013/12/06 03:44:58,407      INFO: tsApplication.TsApplication:
      "guiTopLevelObjectId" = <"-1">.
2013/12/06 03:44:58,407      INFO: tsApplication.TsApplication:
      "guiMessageRedirect" = <"True">.
2013/12/06 03:44:58,407      INFO: tsApplication.TsApplication:
      "guiMessageFilename" = <"None">.
2013/12/06 03:44:58,407      INFO: tsApplication.TsApplication:
      "guiTopLevelObject" = <"<class '__main__._Communicate'>">.
2013/12/06 03:44:58,407      INFO: tsApplication.TsApplication:
      "guiTopLevelObjectName" = <"frame">.
2013/12/06 03:44:58,407      INFO: tsApplication.TsApplication:
      "guiTopLevelObjectParent" = <"None">.
2013/12/06 03:44:58,407      INFO: tsApplication.TsApplication:
      "guiTopLevelObjectPosition" = <"(-1, -1)">.
2013/12/06 03:44:58,407      INFO: tsApplication.TsApplication:
      "guiTopLevelObjectSize" = <"(-1, -1)">.
2013/12/06 03:44:58,407      INFO: tsApplication.TsApplication:
      "guiTopLevelObjectStatusBar" = <"None">.
2013/12/06 03:44:58,407      INFO: tsApplication.TsApplication:
      "guiTopLevelObjectStyle" = <"570433088">.
2013/12/06 03:44:58,407      INFO: tsApplication.TsApplication:
```

```
"guiTopLevelObjectTitle" = <"Gui_Test_Units">.
2013/12/06 03:44:58,407      INFO: tsApplication.TsApplication:
    "buildTitleVersionDate" = <"test_tsWxWidgets, v2.3.0 (build
06/04/2013)">.
2013/12/06 03:44:58,407      INFO: tsApplication.TsApplication:
    "buildHeader" = <"

test_tsWxWidgets, v2.3.0 (build 06/04/2013)

Author(s): Richard S. Gordon
Copyright (c) 2007-2013 Richard S. Gordon.
    All rights reserved.
GNU General Public License, Version 3, 29 June 2007

Credits:

    tsLibGUI Import & Application Launch Features:
    Copyright (c) 2007-2009 Frederick A. Kier.
        All rights reserved.

    Python Curses Module API & Run Time Library Features:
    Copyright (c) 2001-2013 Python Software Foundation.
        All rights reserved.
    PSF License Agreement for Python 2.7.3 & 3.3.0

    wxWidgets (formerly wxWindows) & wxPython API Features:
    Copyright (c) 1992-2008 Julian Smart, Robert Roebling,
        Vadim Zeitlin and other members of the
        wxWidgets team.
        All rights reserved.
    wxWindows Library License

    nCurses API & Run Time Library Features:
    Copyright (c) 1998-2011 Free Software Foundation, Inc.
        All rights reserved.
    GNU General Public License, Version 3, 29 June 2007
">.
2013/12/06 03:44:58,407      INFO: tsApplication.TsApplication:
    "runTimeEntryPoint" = <"<function EntryPoint at 0x7fcf464c>">.
```



```

2013/12/06 03:44:58,407      INFO: dir(self.parent)=[ 'Wrapper', '_App',
'_Logs', '_TheAssignedLogger', '_TheAssignedLogger', '__class__',
'__delattr__', '__dict__', '__doc__', '__format__', '__getattr__',
'__hash__', '__init__', '__module__', '__new__', '__reduce__',
'__reduce_ex__', '__repr__', '__setattr__', '__sizeof__', '__str__',
'__subclasshook__', '__weakref__', 'applicationStyle', 'args', 'buildAuthors',
'buildCopyright', 'buildCredits', 'buildDate', 'buildHeader', 'buildLicense',
'buildPurpose', 'buildTitle', 'buildTitleVersionDate', 'buildVersion',
'callersExceptionHandler', 'createLog', 'currentTime',
'enableDefaultCommandLineParser', 'getApp', 'getAssignedLogger',
'getLaunchSettings', 'getLog', 'getRunTimeTitle',
'getRunTimeTitleVersionDate', 'guiMessageFilename', 'guiMessageRedirect',
'guiModeLauncher', 'guiRequired', 'guiTopLevelObject', 'guiTopLevelObjectId',
'guiTopLevelObjectName', 'guiTopLevelObjectParent',
'guiTopLevelObjectPosition', 'guiTopLevelObjectSize',
'guiTopLevelObjectStatusBar', 'guiTopLevelObjectStyle',
'guiTopLevelObjectTitle', 'logger', 'logs', 'options', 'registerBuildAuthors',
'registerBuildCopyright', 'registerBuildCredits', 'registerBuildDate',
'registerBuildHeader', 'registerBuildLicense', 'registerBuildPurpose',
'registerBuildTitle', 'registerBuildTitleVersionDate', 'registerBuildVersion',
'registerGuiMessageFilename', 'registerGuiMessageRedirect',
'registerGuiRequired', 'registerGuiTopLevelObject',
'registerGuiTopLevelObjectId', 'registerGuiTopLevelObjectName',
'registerGuiTopLevelObjectParent', 'registerGuiTopLevelObjectPosition',
'registerGuiTopLevelObjectSize', 'registerGuiTopLevelObjectStatusBar',
'registerGuiTopLevelObjectStyle', 'registerGuiTopLevelObjectTitle',
'registerInstantiationSettings', 'registerLogs', 'registerRunTimeEntryPoint',
'runMainApplication', 'runTimeEntryPoint', 'runTimeTitle',
'runTimeTitleVersionDate', 'runTimeTrap', 'shutdownTime', 'startupTime',
'sysArgv', 'tsAppArgs', 'tsAppKw']
2013/12/06 03:44:58,417      DEBUG: Begin GraphicalTextUserInterface
(0x7F6C60AC) for Display.
2013/12/06 03:44:58,417      DEBUG: Begin Curses Start.
2013/12/06 03:44:58,417      DEBUG: stdscr = <_curses.curses window object
at 0x7ff921f0>
2013/12/06 03:44:58,417      DEBUG: stdscrGeometry = (0, 0, 80, 35)
2013/12/06 03:44:58,417      DEBUG: stdscrGeometryPixels = (0, 0, 640, 420)
2013/12/06 03:44:58,417      DEBUG: termname = xterm
2013/12/06 03:44:58,417      DEBUG: longname =
2013/12/06 03:44:58,417      DEBUG: foreground = black
2013/12/06 03:44:58,417      DEBUG: background = white
2013/12/06 03:44:58,417      DEBUG: mmask = 536870911
2013/12/06 03:44:58,417      DEBUG: has_mouse = True
2013/12/06 03:44:58,417      DEBUG: MouseButtonCodes = {4096: 'button 3
clicked', 1: 'button 1 released', 2: 'button 1 pressed', 4: 'button 1
clicked', 524288: 'button 4 triple clicked', 8: 'button 1 double clicked',
128: 'button 2 clicked', 134217728: 'button alt', 256: 'button 2 double
clicked', 16: 'button 1 triple clicked', 512: 'button 2 triple clicked',
33554432: 'button ctrl', 67108864: 'button shift', 32: 'button 2 released',
131072: 'button 4 clicked', 262144: 'button 4 double clicked', 1024: 'button 3
released', 8192: 'button 3 double clicked', 16384: 'button 3 triple clicked',
32768: 'button 4 released', 64: 'button 2 pressed', 2048: 'button 3 pressed',
65536: 'button 4 pressed', 'name': 'MouseButtonCodes'}
2013/12/06 03:44:58,417      DEBUG: has_colors = True
2013/12/06 03:44:58,417      DEBUG: curses_colors = 8
2013/12/06 03:44:58,417      DEBUG: curses_color_pairs = 64

```

```
2013/12/06 03:44:58,417      DEBUG:      can_change_color = False
2013/12/06 03:44:58,417      DEBUG:      Begin tsInstallDefaultColorDataBase
2013/12/06 03:44:58,417      DEBUG: installed standard ColorDataBase = {'blue':
4, 'name': 'ColorDataBase', 'yellow': 3, 'green': 2, 'cyan': 6, 'black': 0,
'magenta': 5, 'white': 7, 'red': 1}
2013/12/06 03:44:58,417      DEBUG: installed ColorDataBaseID = {0: 'black', 1:
'red', 2: 'green', 3: 'yellow', 4: 'blue', 'name': 'ColorDataBaseID', 6:
'cyan', 7: 'white', 5: 'magenta'}
2013/12/06 03:44:58,417      DEBUG: installed ColorDataBaseRGB = {'blue': (0,
0, 202), 'name': 'ColorDataBaseRGB', 'yellow': (202, 202, 0), 'green': (0,
202, 0), 'cyan': (0, 202, 202), 'black': (0, 0, 0), 'magenta': (202, 0, 202),
'white': (202, 202, 202), 'red': (202, 0, 0)}
2013/12/06 03:44:58,417      DEBUG: installed ColorSubstitutionDataBase =
{'cadet blue': 'blue', 'sea green': 'green', 'gold': 'yellow', 'firebrick':
'red', 'medium goldenrod': 'yellow', 'violet': 'magenta', 'steel blue':
'blue', 'maroon': 'red', 'sienna': 'red', 'dark slate blue': 'blue', 'khaki':
'yellow', 'medium turquoise': 'cyan', 'sky blue': 'cyan', 'navy': 'blue',
'light blue': 'blue', 'lime green': 'green', 'magenta': 'magenta', 'blue
violet': 'blue', 'orchid': 'magenta', 'blue': 'blue', 'violet red': 'red',
'medium aquamarine': 'cyan', 'medium violet red': 'red', 'medium slate blue':
'blue', 'purple': 'red', 'dark turquoise': 'cyan', 'thistle': 'black', 'light
steel blue': 'blue', 'black': 'black', 'medium spring green': 'green', 'indian
red': 'red', 'aquamarine': 'cyan', 'white': 'white', 'medium sea green':
'green', 'red': 'red', 'brown': 'yellow', 'turquoise': 'cyan', 'dim gray':
'black', 'wheat': 'yellow', 'yellow green': 'yellow', 'medium orchid':
'magenta', 'salmon': 'red', 'dark gray': 'black', 'orange': 'red', 'yellow':
'yellow', 'spring green': 'green', 'dark slate gray': 'black', 'dark olive
green': 'green', 'cyan': 'cyan', 'green yellow': 'green', 'orange red': 'red',
'tan': 'yellow', 'midnight blue': 'blue', 'gray': 'black', 'cornflower blue':
'blue', 'goldenrod': 'cyan', 'pink': 'red', 'name': 'colorSubstitutionMap',
'coral': 'red', 'medium forest green': 'green', 'medium blue': 'blue', 'forest
green': 'green', 'dark orchid': 'magenta', 'slate blue': 'blue', 'pale green':
'green', 'green': 'green', 'light gray': 'black', 'plum': 'magenta', 'dark
green': 'green'}
```

```

2013/12/06 03:44:58,417      DEBUG: installed standard ColorDataBasePairID =
{'ColorNumbersFromPairNumbers': {0: (0, 0), 1: (1, 0), 2: (2, 0), 3: (3, 0),
4: (4, 0), 5: (5, 0), 6: (6, 0), 7: (7, 0), 8: (0, 1), 9: (1, 1), 10: (2, 1),
11: (3, 1), 12: (4, 1), 13: (5, 1), 14: (6, 1), 15: (7, 1), 16: (0, 2), 17:
(1, 2), 18: (2, 2), 19: (3, 2), 20: (4, 2), 21: (5, 2), 22: (6, 2), 23: (7,
2), 24: (0, 3), 25: (1, 3), 26: (2, 3), 27: (3, 3), 28: (4, 3), 29: (5, 3),
30: (6, 3), 31: (7, 3), 32: (0, 4), 33: (1, 4), 34: (2, 4), 35: (3, 4), 36:
(4, 4), 37: (5, 4), 38: (6, 4), 39: (7, 4), 40: (0, 5), 41: (1, 5), 42: (2,
5), 43: (3, 5), 44: (4, 5), 45: (5, 5), 46: (6, 5), 47: (7, 5), 48: (0, 6),
49: (1, 6), 50: (2, 6), 51: (3, 6), 52: (4, 6), 53: (5, 6), 54: (6, 6), 55:
(7, 6), 56: (0, 7), 57: (1, 7), 58: (2, 7), 59: (3, 7), 60: (4, 7), 61: (5,
7), 62: (6, 7), 63: (7, 7), 'name': 'ColorNumbersFromPairNumbers'},
'PairNumbersFromColorNumbers': {(7, 3): 31, (4, 7): 60, (1, 3): 25, (6, 6):
54, (3, 0): 3, (5, 4): 37, (2, 1): 10, (4, 6): 52, (5, 6): 53, (6, 2): 22, (1,
6): 49, (3, 7): 59, (5, 1): 13, (0, 3): 24, (2, 5): 42, (7, 2): 23, (4, 0): 4,
(1, 2): 17, (6, 7): 62, (3, 3): 27, (0, 6): 48, (7, 6): 55, (4, 4): 36, (6,
3): 30, (1, 5): 41, (5, 0): 5, (2, 2): 18, (5, 7): 61, (3, 5): 43, (4, 1): 12,
(1, 1): 9, (6, 4): 38, (3, 2): 19, (0, 0): 0, (3, 6): 51, (2, 7): 58, (7, 1):
15, (4, 5): 44, (0, 4): 32, (6, 0): 6, (1, 4): 33, (7, 7): 63, (5, 5): 45, (7,
5): 47, (2, 3): 26, (0, 7): 56, (4, 2): 20, (1, 0): 1, (6, 5): 46, (5, 3): 29,
(0, 1): 8, (7, 0): 7, 'name': 'PairNumbersFromColorNumbers', (3, 4): 35, (6,
1): 14, (3, 1): 11, (2, 6): 50, (2, 4): 34, (7, 4): 39, (2, 0): 2, (4, 3): 28,
(1, 7): 57, (0, 5): 40, (5, 2): 21, (0, 2): 16}, 'name':
'ColorDataBasePairID'}
2013/12/06 03:44:58,417      DEBUG:      End tsInstallDefaultColorDataBase
2013/12/06 03:44:58,427      DEBUG:      End Curses Start.
2013/12/06 03:44:58,427      NOTICE: start: wxThemeToUse:
currentDirectory=/cygdrive/d/WR/SoftwareGadgetry-Dev/Python-2x/tsWxGTUI;
theSplashScreenPath=./tsLibGUI/tsWxPkg/src/;
theSplashScreenFileName=theSplashScreen.img
2013/12/06 03:44:58,427      NOTICE: start: platformSuffix=_80_x_35_cygwin_nt-
6.1.img
2013/12/06 03:44:58,427      NOTICE: start:
bitmapImageFileName=./tsLibGUI/tsWxPkg/src/theSplashScreen_80_x_35_cygwin_nt-
6.1.img
2013/12/06 03:44:58,427      DEBUG: type(bitmapID)=<type '_curses.curses
window'>
2013/12/06 03:45:38,257      WARNING: Received SIGINT
2013/12/06 03:45:38,257      DEBUG:      Begin Curses Stop.
2013/12/06 03:45:38,347      ERROR: Splash Screen Other Error: 'User Interface
Exception: Command Line Operation Not Valid; Command: "stty sane". [ExitCode
#133]'
2013/12/06 03:45:38,347      DEBUG: End GraphicalTextUserInterface (0x7F6C60AC)
for Display.
2013/12/06 03:45:38,347      DEBUG: Begin GraphicalTextUserInterface
(0x7F6B1ACC) for PyApp.
2013/12/06 03:45:38,347      DEBUG:      Begin Curses Start.
2013/12/06 03:45:38,347      DEBUG:      stdscr = <_curses.curses window object
at 0x7ff92290>
2013/12/06 03:45:38,347      DEBUG:      stdscrGeometry = (0, 0, 80, 35)
2013/12/06 03:45:38,347      DEBUG:      stdscrGeometryPixels = (0, 0, 640, 420)
2013/12/06 03:45:38,347      DEBUG:      termname = xterm
2013/12/06 03:45:38,347      DEBUG:      longname =
2013/12/06 03:45:38,347      DEBUG:      foreground = black
2013/12/06 03:45:38,347      DEBUG:      background = white
2013/12/06 03:45:38,347      DEBUG:      mmask = 536870911

```

```
2013/12/06 03:45:38,347      DEBUG:      has_mouse = True
2013/12/06 03:45:38,347      DEBUG:      MouseButtonCodes = {4096: 'button 3
clicked', 1: 'button 1 released', 2: 'button 1 pressed', 4: 'button 1
clicked', 524288: 'button 4 triple clicked', 8: 'button 1 double clicked',
128: 'button 2 clicked', 134217728: 'button alt', 256: 'button 2 double
clicked', 16: 'button 1 triple clicked', 512: 'button 2 triple clicked',
33554432: 'button ctrl', 67108864: 'button shift', 32: 'button 2 released',
131072: 'button 4 clicked', 262144: 'button 4 double clicked', 1024: 'button 3
released', 8192: 'button 3 double clicked', 16384: 'button 3 triple clicked',
32768: 'button 4 released', 64: 'button 2 pressed', 2048: 'button 3 pressed',
65536: 'button 4 pressed', 'name': 'MouseButtonCodes'}
2013/12/06 03:45:38,347      DEBUG:      has_colors = True
2013/12/06 03:45:38,347      DEBUG:      curses_colors = 8
2013/12/06 03:45:38,347      DEBUG:      curses_color_pairs = 64
2013/12/06 03:45:38,347      DEBUG:      can_change_color = False
2013/12/06 03:45:38,347      DEBUG:      Begin tsInstallDefaultColorDataBase
2013/12/06 03:45:38,347      DEBUG: installed standard ColorDataBase = {'blue':
4, 'name': 'ColorDataBase', 'yellow': 3, 'green': 2, 'cyan': 6, 'black': 0,
'magenta': 5, 'white': 7, 'red': 1}
2013/12/06 03:45:38,347      DEBUG: installed ColorDataBaseID = {0: 'black', 1:
'red', 2: 'green', 3: 'yellow', 4: 'blue', 'name': 'ColorDataBaseID', 6:
'cyan', 7: 'white', 5: 'magenta'}
2013/12/06 03:45:38,347      DEBUG: installed ColorDataBaseRGB = {'blue': (0,
0, 202), 'name': 'ColorDataBaseRGB', 'yellow': (202, 202, 0), 'green': (0,
202, 0), 'cyan': (0, 202, 202), 'black': (0, 0, 0), 'magenta': (202, 0, 202),
'white': (202, 202, 202), 'red': (202, 0, 0)}
2013/12/06 03:45:38,347      DEBUG: installed ColorSubstitutionDataBase =
{'cadet blue': 'blue', 'sea green': 'green', 'gold': 'yellow', 'firebrick':
'red', 'medium goldenrod': 'yellow', 'violet': 'magenta', 'steel blue':
'blue', 'maroon': 'red', 'sienna': 'red', 'dark slate blue': 'blue', 'khaki':
'yellow', 'medium turquoise': 'cyan', 'sky blue': 'cyan', 'navy': 'blue',
'light blue': 'blue', 'lime green': 'green', 'magenta': 'magenta', 'blue
violet': 'blue', 'orchid': 'magenta', 'blue': 'blue', 'violet red': 'red',
'medium aquamarine': 'cyan', 'medium violet red': 'red', 'medium slate blue':
'blue', 'purple': 'red', 'dark turquoise': 'cyan', 'thistle': 'black', 'light
steel blue': 'blue', 'black': 'black', 'medium spring green': 'green', 'indian
red': 'red', 'aquamarine': 'cyan', 'white': 'white', 'medium sea green':
'green', 'red': 'red', 'brown': 'yellow', 'turquoise': 'cyan', 'dim gray':
'black', 'wheat': 'yellow', 'yellow green': 'yellow', 'medium orchid':
'magenta', 'salmon': 'red', 'dark gray': 'black', 'orange': 'red', 'yellow':
'yellow', 'spring green': 'green', 'dark slate gray': 'black', 'dark olive
green': 'green', 'cyan': 'cyan', 'green yellow': 'green', 'orange red': 'red',
'tan': 'yellow', 'midnight blue': 'blue', 'gray': 'black', 'cornflower blue':
'blue', 'goldenrod': 'cyan', 'pink': 'red', 'name': 'colorSubstitutionMap',
'coral': 'red', 'medium forest green': 'green', 'medium blue': 'blue', 'forest
green': 'green', 'dark orchid': 'magenta', 'slate blue': 'blue', 'pale green':
'green', 'green': 'green', 'light gray': 'black', 'plum': 'magenta', 'dark
green': 'green'}
```

```

2013/12/06 03:45:38,347      DEBUG: installed standard ColorDataBasePairID =
{'ColorNumbersFromPairNumbers': {0: (0, 0), 1: (1, 0), 2: (2, 0), 3: (3, 0),
4: (4, 0), 5: (5, 0), 6: (6, 0), 7: (7, 0), 8: (0, 1), 9: (1, 1), 10: (2, 1),
11: (3, 1), 12: (4, 1), 13: (5, 1), 14: (6, 1), 15: (7, 1), 16: (0, 2), 17:
(1, 2), 18: (2, 2), 19: (3, 2), 20: (4, 2), 21: (5, 2), 22: (6, 2), 23: (7,
2), 24: (0, 3), 25: (1, 3), 26: (2, 3), 27: (3, 3), 28: (4, 3), 29: (5, 3),
30: (6, 3), 31: (7, 3), 32: (0, 4), 33: (1, 4), 34: (2, 4), 35: (3, 4), 36:
(4, 4), 37: (5, 4), 38: (6, 4), 39: (7, 4), 40: (0, 5), 41: (1, 5), 42: (2,
5), 43: (3, 5), 44: (4, 5), 45: (5, 5), 46: (6, 5), 47: (7, 5), 48: (0, 6),
49: (1, 6), 50: (2, 6), 51: (3, 6), 52: (4, 6), 53: (5, 6), 54: (6, 6), 55:
(7, 6), 56: (0, 7), 57: (1, 7), 58: (2, 7), 59: (3, 7), 60: (4, 7), 61: (5,
7), 62: (6, 7), 63: (7, 7), 'name': 'ColorNumbersFromPairNumbers'},
'PairNumbersFromColorNumbers': {(7, 3): 31, (4, 7): 60, (1, 3): 25, (6, 6):
54, (3, 0): 3, (5, 4): 37, (2, 1): 10, (4, 6): 52, (5, 6): 53, (6, 2): 22, (1,
6): 49, (3, 7): 59, (5, 1): 13, (0, 3): 24, (2, 5): 42, (7, 2): 23, (4, 0): 4,
(1, 2): 17, (6, 7): 62, (3, 3): 27, (0, 6): 48, (7, 6): 55, (4, 4): 36, (6,
3): 30, (1, 5): 41, (5, 0): 5, (2, 2): 18, (5, 7): 61, (3, 5): 43, (4, 1): 12,
(1, 1): 9, (6, 4): 38, (3, 2): 19, (0, 0): 0, (3, 6): 51, (2, 7): 58, (7, 1):
15, (4, 5): 44, (0, 4): 32, (6, 0): 6, (1, 4): 33, (7, 7): 63, (5, 5): 45, (7,
5): 47, (2, 3): 26, (0, 7): 56, (4, 2): 20, (1, 0): 1, (6, 5): 46, (5, 3): 29,
(0, 1): 8, (7, 0): 7, 'name': 'PairNumbersFromColorNumbers', (3, 4): 35, (6,
1): 14, (3, 1): 11, (2, 6): 50, (2, 4): 34, (7, 4): 39, (2, 0): 2, (4, 3): 28,
(1, 7): 57, (0, 5): 40, (5, 2): 21, (0, 2): 16}, 'name':
'ColorDataBasePairID'}
2013/12/06 03:45:38,347      DEBUG:      End tsInstallDefaultColorDataBase
2013/12/06 03:45:38,357      DEBUG:      End Curses Start.
2013/12/06 03:45:38,357      NOTICE: start: wxThemeToUse:
currentDirectory=/cygdrive/d/WR/SoftwareGadgetry-Dev/Python-2x/tsWxGTUI;
theSplashScreenPath=./tsLibGUI/tsWxPkg/src/;
theSplashScreenFileName=theSplashScreen.img
2013/12/06 03:45:38,357      NOTICE: start: platformSuffix=_80_x_35_cygwin_nt-
6.1.img
2013/12/06 03:45:38,357      NOTICE: start:
bitmapImageFileName=./tsLibGUI/tsWxPkg/src/theSplashScreen_80_x_35_cygwin_nt-
6.1.img
2013/12/06 03:45:38,357      DEBUG: type(bitmapID)=<type '_curses.curses
window'>
2013/12/06 03:45:39,617      WARNING: Received SIGINT
2013/12/06 03:45:39,617      DEBUG:      Begin Curses Stop.
2013/12/06 03:45:39,617      ERROR: Splash Screen Other Error: 'User Interface
Exception: Command Line Operation Not Valid; Command: "stty sane". [ExitCode
#133]'
2013/12/06 03:45:39,617      DEBUG: End GraphicalTextUserInterface (0x7F6B1ACC)
for PyApp.

```

11.1.3.1.3 TerminalRunTimeEnvironment.log

tsWxGraphicalTextUserInterface, v2.8.0 (build 12/02/2013)

Author(s): Richard S. Gordon
Copyright (c) 2007-2013 Richard S. Gordon.
All rights reserved.
GNU General Public License, Version 3, 29 June 2007

Credits:

tsLibGUI Import & Application Launch Features:
Copyright (c) 2007-2009 Frederick A. Kier.
All rights reserved.

Python Curses Module API & Run Time Library Features:
Copyright (c) 2001-2013 Python Software Foundation.
All rights reserved.
PSF License Agreement for Python 2.7.3 & 3.3.0

wxWidgets (formerly wxWindows) & wxPython API Features:
Copyright (c) 1992-2008 Julian Smart, Robert Roebling,
Vadim Zeitlin and other members of the
wxWidgets team.
All rights reserved.
wxWindows Library License

nCurses character-mode Terminal Control Library
for Unix-like systems and API Features:
Copyright (c) 1998-2004, 2006 Free Software
Foundation, Inc.
All rights reserved.
nCurses README, v 1.23 2006/04/22

2013/12/06 03:45:38,347 - Started logging to file "./logs/2013-12-06-at-03-44-55/TerminalRunTimeEnvironment.log"

----- Begin "CursesDataBase" at level 0 -----

 BackgroundColor = white

----- Begin "BuiltinPaletteRGB" at level 1 -----

 black = (0, 0, 0)
 blue = (0, 0, 202)
 cyan = (0, 202, 202)
 green = (0, 202, 0)
 magenta = (202, 0, 202)
 name = BuiltinPaletteRGB
 red = (202, 0, 0)
 white = (202, 202, 202)
 yellow = (202, 202, 0)

----- End "BuiltinPaletteRGB" at level 1 -----

CanChangeColor = 0 (0x0)

----- Begin "ColorDataBase" at level 1 -----

black = 0 (0x0)
blue = 4 (0x4)
cyan = 6 (0x6)
green = 2 (0x2)
magenta = 5 (0x5)
name = ColorDataBase
red = 1 (0x1)
white = 7 (0x7)
yellow = 3 (0x3)

----- End "ColorDataBase" at level 1 -----

----- Begin "ColorDataBaseID" at level 1 -----

0 = black
1 = red
2 = green
3 = yellow
4 = blue
5 = magenta
6 = cyan
7 = white
name = ColorDataBaseID

----- End "ColorDataBaseID" at level 1 -----

----- Begin "ColorDataBasePairID" at level 1 -----

----- Begin "ColorNumbersFromPairNumbers" at level 2 -----

0 = (0, 0)
1 = (1, 0)
2 = (2, 0)
3 = (3, 0)
4 = (4, 0)
5 = (5, 0)
6 = (6, 0)
7 = (7, 0)
8 = (0, 1)
9 = (1, 1)
10 = (2, 1)
11 = (3, 1)
12 = (4, 1)
13 = (5, 1)
14 = (6, 1)
15 = (7, 1)
16 = (0, 2)

```
17 = (1, 2)
18 = (2, 2)
19 = (3, 2)
20 = (4, 2)
21 = (5, 2)
22 = (6, 2)
23 = (7, 2)
24 = (0, 3)
25 = (1, 3)
26 = (2, 3)
27 = (3, 3)
28 = (4, 3)
29 = (5, 3)
30 = (6, 3)
31 = (7, 3)
32 = (0, 4)
33 = (1, 4)
34 = (2, 4)
35 = (3, 4)
36 = (4, 4)
37 = (5, 4)
38 = (6, 4)
39 = (7, 4)
40 = (0, 5)
41 = (1, 5)
42 = (2, 5)
43 = (3, 5)
44 = (4, 5)
45 = (5, 5)
46 = (6, 5)
47 = (7, 5)
48 = (0, 6)
49 = (1, 6)
50 = (2, 6)
51 = (3, 6)
52 = (4, 6)
53 = (5, 6)
54 = (6, 6)
55 = (7, 6)
56 = (0, 7)
57 = (1, 7)
58 = (2, 7)
59 = (3, 7)
60 = (4, 7)
61 = (5, 7)
62 = (6, 7)
63 = (7, 7)
name = ColorNumbersFromPairNumbers

----- End    "ColorNumbersFromPairNumbers" at level 2 -----

----- Begin "PairNumbersFromColorNumbers" at level 2 -----

(0, 0) = 0 (0x0)
(0, 1) = 8 (0x8)
```


(0, 2) = 16 (0x10)
(0, 3) = 24 (0x18)
(0, 4) = 32 (0x20)
(0, 5) = 40 (0x28)
(0, 6) = 48 (0x30)
(0, 7) = 56 (0x38)
(1, 0) = 1 (0x1)
(1, 1) = 9 (0x9)
(1, 2) = 17 (0x11)
(1, 3) = 25 (0x19)
(1, 4) = 33 (0x21)
(1, 5) = 41 (0x29)
(1, 6) = 49 (0x31)
(1, 7) = 57 (0x39)
(2, 0) = 2 (0x2)
(2, 1) = 10 (0xA)
(2, 2) = 18 (0x12)
(2, 3) = 26 (0x1A)
(2, 4) = 34 (0x22)
(2, 5) = 42 (0x2A)
(2, 6) = 50 (0x32)
(2, 7) = 58 (0x3A)
(3, 0) = 3 (0x3)
(3, 1) = 11 (0xB)
(3, 2) = 19 (0x13)
(3, 3) = 27 (0x1B)
(3, 4) = 35 (0x23)
(3, 5) = 43 (0x2B)
(3, 6) = 51 (0x33)
(3, 7) = 59 (0x3B)
(4, 0) = 4 (0x4)
(4, 1) = 12 (0xC)
(4, 2) = 20 (0x14)
(4, 3) = 28 (0x1C)
(4, 4) = 36 (0x24)
(4, 5) = 44 (0x2C)
(4, 6) = 52 (0x34)
(4, 7) = 60 (0x3C)
(5, 0) = 5 (0x5)
(5, 1) = 13 (0xD)
(5, 2) = 21 (0x15)
(5, 3) = 29 (0x1D)
(5, 4) = 37 (0x25)
(5, 5) = 45 (0x2D)
(5, 6) = 53 (0x35)
(5, 7) = 61 (0x3D)
(6, 0) = 6 (0x6)
(6, 1) = 14 (0xE)
(6, 2) = 22 (0x16)
(6, 3) = 30 (0x1E)
(6, 4) = 38 (0x26)
(6, 5) = 46 (0x2E)
(6, 6) = 54 (0x36)
(6, 7) = 62 (0x3E)
(7, 0) = 7 (0x7)
(7, 1) = 15 (0xF)

```
(7, 2) = 23 (0x17)
(7, 3) = 31 (0x1F)
(7, 4) = 39 (0x27)
(7, 5) = 47 (0x2F)
(7, 6) = 55 (0x37)
(7, 7) = 63 (0x3F)
    name = PairNumbersFromColorNumbers

----- End    "PairNumbersFromColorNumbers" at level 2 -----

        name = ColorDataBasePairID

----- End    "ColorDataBasePairID" at level 1 -----

----- Begin "ColorDataBaseRGB" at level 1 -----

    black = (0, 0, 0)
    blue = (0, 0, 202)
    cyan = (0, 202, 202)
    green = (0, 202, 0)
    magenta = (202, 0, 202)
    name = ColorDataBaseRGB
    red = (202, 0, 0)
    white = (202, 202, 202)
    yellow = (202, 202, 0)

----- End    "ColorDataBaseRGB" at level 1 -----

----- Begin "colorSubstitutionMap" at level 1 -----

    aquamarine = cyan
    black = black
    blue = blue
    blue violet = blue
    brown = yellow
    cadet blue = blue
    coral = red
    cornflower blue = blue
    cyan = cyan
    dark gray = black
    dark green = green
    dark olive green = green
    dark orchid = magenta
    dark slate blue = blue
    dark slate gray = black
    dark turquoise = cyan
    dim gray = black
    firebrick = red
    forest green = green
    gold = yellow
    goldenrod = cyan
    gray = black
    green = green
    green yellow = green
```

```

        indian red = red
            khaki = yellow
        light blue = blue
        light gray = black
    light steel blue = blue
        lime green = green
            magenta = magenta
            maroon = red
    medium aquamarine = cyan
        medium blue = blue
    medium forest green = green
        medium goldenrod = yellow
            medium orchid = magenta
        medium sea green = green
        medium slate blue = blue
    medium spring green = green
        medium turquoise = cyan
    medium violet red = red
        midnight blue = blue
            name = colorSubstitutionMap
            navy = blue
            orange = red
        orange red = red
            orchid = magenta
    pale green = green
        pink = red
        plum = magenta
        purple = red
            red = red
        salmon = red
        sea green = green
        sienna = red
        sky blue = cyan
        slate blue = blue
    spring green = green
        steel blue = blue
            tan = yellow
            thistle = black
        turquoise = cyan
            violet = magenta
    violet red = red
        wheat = yellow
        white = white
        yellow = yellow
    yellow green = yellow

```

----- End "colorSubstitutionMap" at level 1 -----

```

    CursesColorPairs = 64 (0x40)
    CursesColors = 8 (0x8)

```

----- Begin "CursesPanels" at level 1 -----

```

    name = CursesPanels

```

----- End "CursesPanels" at level 1 -----

```
ForegroundColor = black
  HasColors = 1 (0x1)
  HasDisplay = 1 (0x1)
  HasKeyboard = 1 (0x1)
  HasLogger = 1 (0x1)
  HasMouse = 1 (0x1)
  HostOS = CYGWIN_NT-6.1
  LongName =
  Mmask = 536870911 (0x1FFFFFFF)
```

```
----- Begin "MouseButtonCodes" at level 1 -----
```

```
    1 = button 1 released
    2 = button 1 pressed
    4 = button 1 clicked
    8 = button 1 double clicked
   16 = button 1 triple clicked
   32 = button 2 released
   64 = button 2 pressed
  128 = button 2 clicked
  256 = button 2 double clicked
  512 = button 2 triple clicked
 1024 = button 3 released
 2048 = button 3 pressed
 4096 = button 3 clicked
 8192 = button 3 double clicked
16384 = button 3 triple clicked
32768 = button 4 released
65536 = button 4 pressed
131072 = button 4 clicked
262144 = button 4 double clicked
524288 = button 4 triple clicked
33554432 = button ctrl
67108864 = button shift
134217728 = button alt
    name = MouseButtonCodes
```

```
----- End   "MouseButtonCodes" at level 1 -----
```

```
    PythonVersion = Python-2.7.3
      Stdscr = <_curses.curses window object at 0x7ff92290>
      StdscrGeometry = (0, 0, 80, 35)
      StdscrGeometryPixels = (0, 0, 640, 420)
      TermName = xterm
      name = CursesDataBase
```

```
----- End   "CursesDataBase" at level 0 -----
```

```
----- Begin "WindowDataBase" at level 0 -----
```

```
----- Begin "AcceleratorKeysByEarliestAssignedId" at level 1 -----
```

```
    name = AcceleratorKeysByEarliestAssignedId
```

```
----- End    "AcceleratorKeysByEarliestAssignedId" at level 1 -----

----- Begin "AcceleratorTableByAssignedId" at level 1 -----
    name = AcceleratorTableByAssignedId
----- End    "AcceleratorTableByAssignedId" at level 1 -----

----- Begin "EventAssociationsByEarliestAssignedId" at level 1 -----
    name = EventAssociationsByEarliestAssignedId
----- End    "EventAssociationsByEarliestAssignedId" at level 1 -----

----- Begin "KeyboardInputRecipients" at level 1 -----
    lifoList = []
    name = KeyboardInputRecipients
----- End    "KeyboardInputRecipients" at level 1 -----

----- Begin "TheWindows" at level 1 -----
    name = TheWindows
    windowIndex = -1 (0x-1)
----- End    "TheWindows" at level 1 -----

----- Begin "TopLevelWindows" at level 1 -----
    name = TopLevelWindows
----- End    "TopLevelWindows" at level 1 -----

----- Begin "WindowHandles" at level 1 -----
    name = WindowHandles
----- End    "WindowHandles" at level 1 -----

----- Begin "WindowTopLevelAncestors" at level 1 -----
    name = WindowTopLevelAncestors
----- End    "WindowTopLevelAncestors" at level 1 -----
    WindowTopLevelTasks = []
```

```
----- Begin "WindowsByAssignedId" at level 1 -----
    name = WindowsByAssignedId
----- End   "WindowsByAssignedId" at level 1 -----

----- Begin "WindowsByHandle" at level 1 -----
    name = WindowsByHandle
----- End   "WindowsByHandle" at level 1 -----

----- Begin "WindowsById" at level 1 -----
    name = WindowsById
----- End   "WindowsById" at level 1 -----

----- Begin "WindowsByName" at level 1 -----
    name = WindowsByName
----- End   "WindowsByName" at level 1 -----

----- Begin "WindowsByPanelLayer" at level 1 -----
    name = WindowsByPanelLayer
----- End   "WindowsByPanelLayer" at level 1 -----

----- Begin "WindowsByShowOrder" at level 1 -----

        ----- Begin "AssignedIdByPanelLayer" at level 2 -----
            name = AssignedIdByPanelLayer
        ----- End   "AssignedIdByPanelLayer" at level 2 -----

            OrderOfShow = []
            OrderOfShowPanelStack = []
            PanelLayer = []

        ----- Begin "PanelStack" at level 2 -----
            name = PanelStack
        ----- End   "PanelStack" at level 2 -----

            name = WindowsByShowOrder
```

----- End "WindowsByShowOrder" at level 1 -----

name = WindowDataBase

----- End "WindowDataBase" at level 0 -----

2013/12/06 03:45:38,347 - Ended logging to file "./logs/2013-12-06-at-03-44-55/TerminalRunTimeEnvironment.log"

Draft